HiT skrift nr 3/2002

Oppretting av polygon

Roy M. Istad

Avdeling for allmenne fag (Bø)
Institutt for informatikk og matematiske fag

Høgskolen i Telemark
Porsgrunn 2002

HiT skrift nr 3/2002

ISBN 82-7206-193-7 (online)
ISBN 82-7206-192-9 (trykt)

ISSN 1503-3767 (online)
ISSN 1501-8539 (trykt)

Høgskolen i Telemark
Postboks 203
3901 Porsgrunn

Telefon 35 57 50 00
Telefaks 35 57 50 01
http://www.hit.no/

Trykk: Kopisenteret. HiT-Bø

Innhold

Sammendrag . . . . . **s 4**

Innledning . . . . . **s 5**

Resultat . . . . . **s 6**
- Polygonopprettingsalgoritme nr 1 **s 6**
- Polygonopprettingsalgoritme nr 2 **s 11**

Aktuelle videreføringer . . . . . **s 17**
- Geografisk orientering av polygon **s 17**
- Arkitektonisk etterjustering av opprettet polygon **s 19**

Vedlegg . . . . . **s 20**
- Pascal-program for algoritmesimulering **s 21**

Referanser . . . . . **s 24**

Sammendrag

Forfatteren ble høsten 2001 kontaktet av et mindre IT-firma her i Telemark med en forespørsel om matematisk assistanse på et problem som oppstod ved digitalisering fra flyfoto. Når man digitaliserer f.eks. en bygning ut fra et flyfoto, så forventes det et resultatet i form av en regelmessig kontur med vinkelrette hjørner. På grunn av ulike unøyaktighetskilder under manuelt digitaliseringsarbeid, blir resultatet svært ofte uregelmessige konturer der vinkelrette hjørner heller er unntaket enn regelen. IT-firmaet ønsket derfor en algoritme som kunne legges inn i programvaren for digitaliseringen, slik at en kan få en automatisert oppretting av den aktuelle konturen (mangekanten, polygonet) som representerer et konkret objekt.

Det var naturlig å definere dette som et internt FoU-prosjekt. Resultatet av arbeidet med å etablere noen aktuelle algoritmer til dette formålet foreligger i denne sluttrapporten fra prosjektet. To av algoritmeforslagene er oversendt og løst drøftet med IT-firmaet, men det er bare å beklage at vedkommende firma ikke har gitt beskjed om en av algoritmene virkelig er blitt tatt i bruk - til tross for eksplisitt forespørsel om slik tilbakemelding.

Problemstillingen og ideene som inngår i algoritmeforslagene er det grunn til å tro kan ha interesse for andre enn bare det aktuelle firmaet. Det er da naturlig og ønskelig å gjøre materialet allment tilgjengelig gjennom skriftserien.

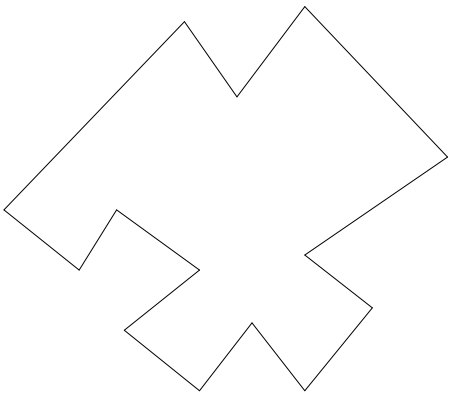
Forfatteren ønsker å takke høgskoledosent Tor Lønnestad for interessante meningsutvekslinger over aktuelle innfallsvinkler til problemet, samt for forslag til videreføringer.

Innledning

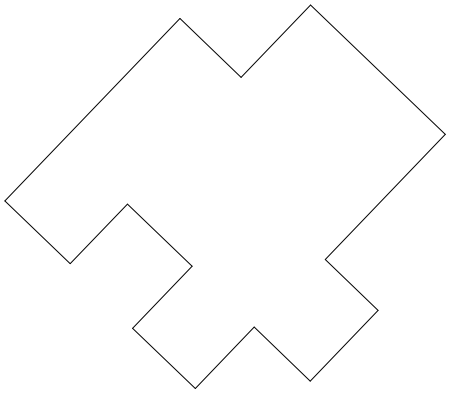
I et konkret GIS-system er det behov for å behandle polygoner (mangekanter) som er fremkommet ved digitalisering av objekt fra flybilder. Dette er objekt som representerer faktiske bygninger som i prinsippet kan oppsøkes ute i terrenget med tanke på kontroll av de registrerte opplysningene. Da er det både et arkitektonisk/estetisk og fagrelevant spørsmål hvorvidt den digitaliserte versjonen av objektet kan brukes videre, eller må rettes opp - i bokstavelig forstand.

Et polygon er her angitt ved et sett av koordinater til hjørnepunktene. Det må være et partallig antall slike hjørnepunkt - og minst fire - for å representere en bygning. Under en manuell digitalisering av et objekt fra flybilde, vil polygonet som representerer objektet være utsatt for flere aktuelle kilder til unøyaktighet. F.eks. kan det være uklart i bildet hvor avgrensningene av objektet egentlig er (hvor veggene til bygningen er) og det kan være vanskelig å presist markere objektets hjørner under digitaliseringsarbeidet med bildet som er gjengitt på en dataskjerm. Selv om et hjørne klart fremkommer på flybildet, og man er presis ved digitalisering av hjørnet, så gjenstår fortsatt pixel-unøyaktigheten under koordinatangivelsen. Punktet som blir "klikket" under skjerm-digitaliseringen blir lagret som nærmest beliggende pixel.

Polygonet som fremkommer etter digitalisering av et konkret objekt, vil typisk være som antydet på fig 1. Man kjenner i hovedtrekk igjen det "korrekte" objektet slik det er vist på fig 2, men i større eller mindre grad avviker det digitaliserte polygonets sidekanter både i retning og lengde i forhold til det "korrekte" objektet.



Figur 1: Digitalisert polygon



Figur 2 : "Korrekt" objekt

Utfordringen ligger i å finne frem til algoritmer som kan rette opp den digitaliserte versjonen av polygonet, slik at man innfrir naturlige krav til det som skal representere en konkret bygning: 1) Vinkelrette hjørner, 2) Korrekt geografisk orientering, 3) Korrekt geografisk plassering og 4) Korrekt arkitektonisk innretning. Dette er ikke en prioritert liste, og den er muligens heller ikke komplett, men den inneholder allerede mer enn store nok utfordringer til aktuelle opprettingsalgoritmer.

Prioriteringen av de forannevnte punktene vil påvirke valg av algoritme, og derfor lanseres tre forskjellige forslag i dette notatet. Det er klart forskjellige krav som stilles i ulike situasjoner. Ta f.eks. det at man enten skal lage et kart i stor målestokk, basert på flybilde over et større terreng med mange bygninger på, eller det at man har flybilde over en eiendom med et par bygninger på og et arkitektkontor skal arbeide videre med digitalisert versjon av en eller to av bygningene.

Resultat

Algoritmeforslag nr 1

Det første algoritmeforslaget er satt opp ut fra et ønske om å gjennomføre minst mulig forhåndsanalyse av datasettet, at den skal inneholde minst mulig regnearbeid (lite omfattende programkode) og at den skal gi et resultat som i første rekke innfrir estetiske krav. Kort sagt så er det viktigste å frembringe et polygon der alle hjørner er vinkelrette, slik at det iallefall innfrir våre forventninger til hvordan bygninger ser ut når de f.eks. er gjengitt på kart.

Det er nødvendig å stille noen krav til datasettet (digitaliseringsarbeidet) for å kunne utforme en algoritme basert på det som er nevnt ovenfor. Den lengste sidekanten på aktuelt objekt må digitaliseres først og deretter må man være spesielt nøyaktig ved digitalisering av annethvert av de påfølgende hjørner.

Felles krav til datasettet (digitalisert polygon) for alle de tre algoritmeforslagene i dette notatet er forøvrig at:

- Det er kun to punkt som er angitt for hver sidekant, ett i hvert hjørne
- Punktene er angitt i hjørnerekkefølge rundt objektet (retningen er ikke vesentlig her)
- Det er angitt et partallig antall, og minst fire, hjørner for objektet.

Dette første algoritmeforslaget er forholdsvis primitivt i minst to henseender. Det forutsettes at dataene er organisert på en spesiell måte, ved at lengste side i objektet skal være digitalisert først. Dernest tillegges hjørnepunktene svært ulik vekt, halparten av dem vil bli oppfattet som absolutt nøyaktige mens den andre halvparten blir sett på som absolutt unøyaktige.

Først til fokuseringen på lengste sidekant. Dette skyldes en vurdering av at retningen til denne trolig vil ha minst usikkerhet. Desto lengre strekk, desto mindre utslag får vi av unøyaktigheten i endepunktene. Dette er under forutsetning av at unøyaktigheten er den samme i alle hjørnepunktene. Første sidekant blir satt som hovedretningen for polygonet, dvs. den skal angi den geografisk orienteringen av objektet. Til hovedretningen hører det selvsagt en sidekant som står vinkelrett på denne, og disse to utvalgte retningene brukes så til justeringen av de øvrige sidekantene.

Så til ulik vektlegging av hjørnepunktene. Dersom det er viktigst å få endret det rent estetiske ved polygonet (rette opp hjørnene), så er det liten arbeidsmengde i det å holde to av tre punkter fast og skyve på det midterste av dem langs hovedretningene til det havner i hjørnet av en rett vinkel (90^o vinkel mellom de to sidekantene - parallelt med hovedretningene - som hører til de tre punktene).

På fig 4 er det gjengitt hvordan algoritmen skyver på det midterste av tre punkt som ligger i rekkefølge på polygonet. De to vinkelrette linjene gjennom det første av de tre punktene, viser de to hovedretningene i opprettingen av polygonet.

Vi skyver altså på annet hvert punkt (merk at det tredje og siste punktet i ei pulje av opprettingsprosessen, blir med i neste pulje som det første av de tre punktene som da skal bearbeides), mens halvparten blir liggende i ro der de lå i utgangspunktet. Dette kan tolkes som at algoritmen oppfatter noen punkter som absolutt nøyaktig angitt, og derfor ikke skal flyttes, mens andre punkter er absolutt unøyaktig angitt og forskyves til en ny posisjon som er uavhengig av deres opprinnelige posisjon. Ny posisjon er entydig bestemt av hovedretningene og beliggenheten av de to (absolutt nøyaktige) nabopunktene. Annethvert punkt blir liggende i ro, og bestemmer dermed også den geografiske posisjoneringen (plasseringen) av objektet

Algoritmen arbeider i to steg:

Steg 1: Bestemmelse av de to hovedretningene for resultatpolygonet.

Steg 2: Oppretting av annethvert hjørne, annethvert beholdes.

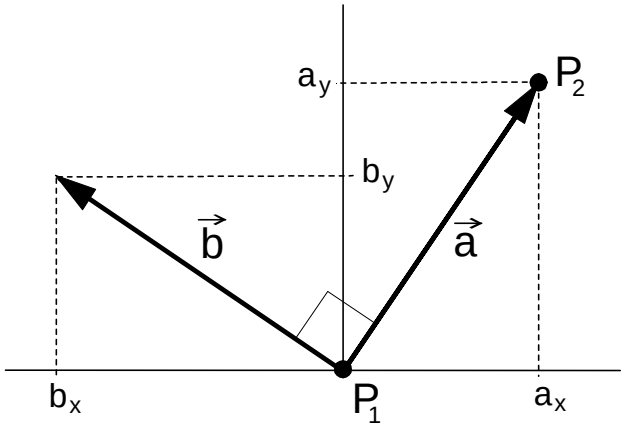
Vi kaller hjørnepunktene for P, og nummerer dem fortløpende i den rekkefølgen de inngår i polygonet: P_1 , P_2 , P_3 , $\cdots$, P_n .
Det er altså et krav at det inngår et partallig antall hjørnepunkt, dvs. at $n = 2k$ og dessuten må vi ha at $n \geq 4$.

Steg 1

I algoritmen bruker vi helt elementær vektorrekning i planet, og alt blir angitt i et rettvinklet koordinatsystem. Hovedretningene til polygonet skal være retningen til lengste sidekant og den som står vinkelrett på.

De to hovedretningene kaller vi heretter basisvektorene $\overrightarrow{a}$ og $\overrightarrow{b}$, der $\overrightarrow{a} = \overrightarrow{P_1P_2}$ er vektoren mellom de to første hjørnepunktene og $\overrightarrow{b}$ er vektoren vinkelrett på $\overrightarrow{a}$. Se fig. 3.

Vektorene $\overrightarrow{a}$ og $\overrightarrow{b}$ utgjør en *ortogonal basis* for planet, dvs. enhver vektor i planet kan skrives som en veid vektorsum (lineær kombinasjon) av $\overrightarrow{a}$ og $\overrightarrow{b}$.



Figur 3: Valg av basisvektorer

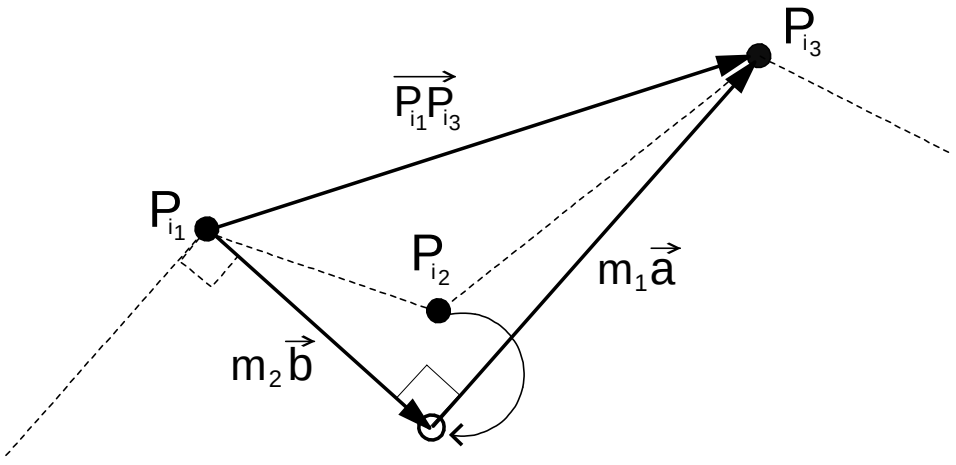
Steg 2

Punktene P_1 og P_2 skal ligge fast. Tre og tre punkt, her kalt P_{i_1} , P_{i_2} og P_{i_3} i rekkefølge rundt polygonet, blir nå satt under behandling.

Som vist på fig 4, blir nye sidelinjer lagt i hovedretningene gjennom P_{i_1} og P_{i_3} , og skjæringspunktet mellom linjene gir koordinatene til det nye hjørnepunktet P_{i_2}. Så behandles neste trio med hjørnepunkter, der det forrige P_{i_3} blir det neste P_{i_1}. Slik vil P_2 , P_3 og P_4 forskyve P_3 , mens P_4 , P_5 og P_6 forskyver P_5 , osv.

Den siste punkttrioen må spesialbehandles. Vi må ut av punktnummereringen og bruke P_{n-1} og P_n sammen med det aller første punktet P_1 til å forskyve P_n. P_n er altså det eneste partallsnummererte punktet som blir forskjøvet, og det fordi P_1 skal ligge i ro.

Fig 4 viser hvordan forskyvningen av et mellomliggende hjørnepunkt P_{i_2} skjer ved at vektoren fra P_{i_1} til P_{i_3} dekomponeres i basisvektorene. Dvs. at vi skriver $\overrightarrow{P_{i_1}P_{i_3}}$ som en lineærkombinasjon av $\overrightarrow{a}$ og $\overrightarrow{b}$, altså $\overrightarrow{P_{i_1}P_{i_3}} = m_1\overrightarrow{a} + m_2\overrightarrow{b}$. Koordinatene til nytt punkt P_{i_2} fremkommer som endepunktet til vektorsummen $\overrightarrow{OP_{i_2}} = \overrightarrow{OP_{i_1}} + m_2\overrightarrow{b}$, der O står for origo $(0, 0)$, slik at koordinatene til aktuell vektor sammenfaller med koordinatene til dens endepunkt.



Figur 4 : Oppretting av et punkttrippel

Beregningene som inngår i algoritmeforslag nr 1.

STEG 1

Bestemmelse av hovedretningene (basisvektorene) $\overrightarrow{a}$ og $\overrightarrow{b}$:

$$P_1 = (p_{1x}\ ,\ p_{1y})\ ,\ \ P_2 = (p_{2x}\ ,\ p_{2y})$$

$$\overrightarrow{a} = (a_x\ ,\ a_y) = \overrightarrow{P_1P_2} = (p_{2x} - p_{1x}\ ,\ p_{2y} - p_{1y})$$

$$\overrightarrow{b} = (b_x\ ,\ b_y) = (-a_y\ ,\ a_x)$$

STEG 2

Flytting av annethvert punkt i det opprinnelige polygonet.

Utfør de følgende regneoperasjonene for punkttripler gitt ved indeksene
$i_1 = 2,\ 4,\ 6,\ \ldots\ ,\ n-2\ ,\quad i_2 = i_1 + 1$ og $i_3 = i_1 + 2$:

$$\overrightarrow{c} = (c_x\ ,\ c_y) = \overrightarrow{P_{i_1}P_{i_3}} = (p_{i_3x} - p_{i_1x}\ ,\ p_{i_3y} - p_{i_1y})$$

$$m_2 = \frac{a_x \cdot c_y\ -\ a_y \cdot c_x}{a_x \cdot b_y\ -\ a_y \cdot b_x}$$

$$P_{i_2} = (p_{i_2x}\ ,\ p_{i_2y}) = \overrightarrow{OP_{i_2}} = \overrightarrow{OP_{i_1}} + m_2 \cdot \overrightarrow{b} = (p_{i_1x} + m_2 \cdot b_x\ ,\ p_{i_1y} + m_2 \cdot b_y)$$

Utfør operasjonene en siste gang med $i_1 = n-1,\ i_2 = n$ og $i_3 = 1$

Merknad

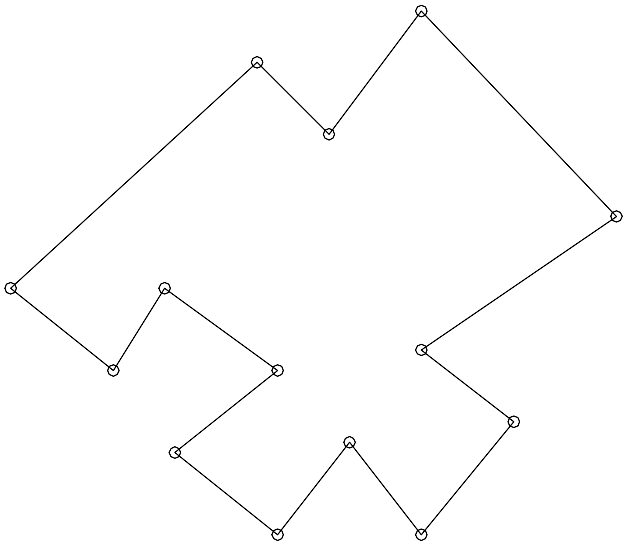
Det er fullt mulig å bestemme m_2 direkte som lengden av den ortogonale projeksjonen av $\overrightarrow{c}$ på $\overrightarrow{b}$:

$$m_2 = \frac{\overrightarrow{c} \cdot \overrightarrow{b}}{\overrightarrow{b} \cdot \overrightarrow{b}} = \frac{b_x \cdot c_x\ +\ b_y \cdot c_y}{b_x \cdot b_x\ +\ b_y \cdot b_y} \qquad \text{(Husk at vi har satt } b_x = -a_y \text{ og } b_y = a_x\text{)}$$

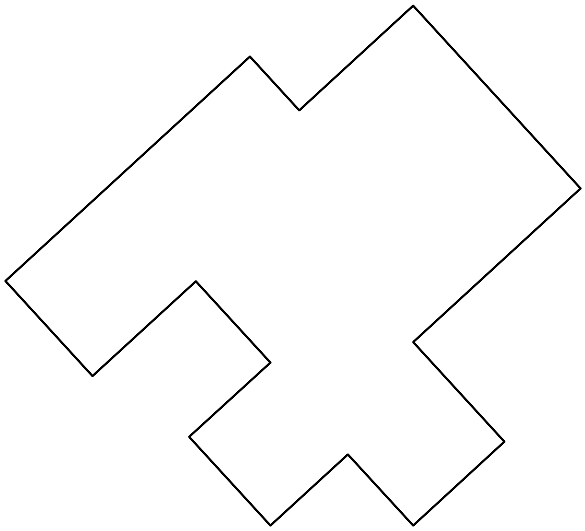
Vi ser at dersom vi hadde normert basisvektorene ($\overrightarrow{a}$ og $\overrightarrow{b}$ begge omregnet til lengde 1), så hadde regnearbeidet for å finne m_2 blitt litt enklere. Dette poenget kommer vi tilbake til under algoritmeforslag nr 2.

Eksempel på polygon opprettet ved algoritme nr 1.

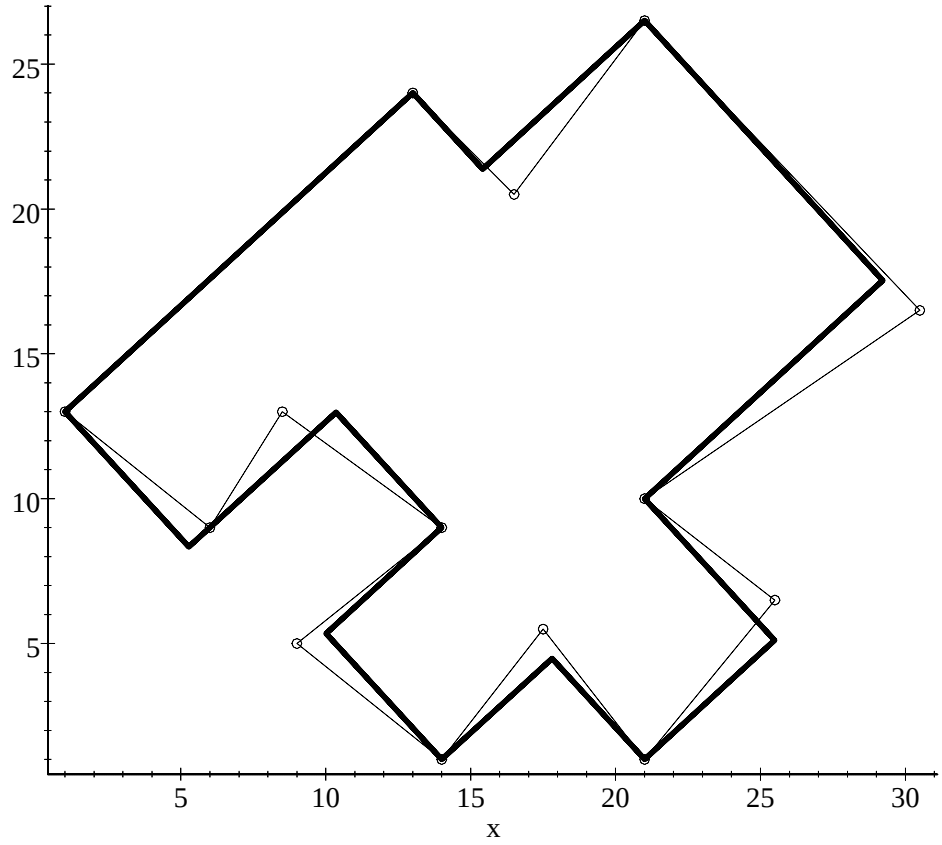
På fig 5 har vi valgt et tilfeldig eksempel på et polygon som kan tenkes å representere et digitalisert objekt (f.eks. en bygning). Første algoritmeforslag er blitt anvendt på dette polygonet, og resultatet er gjengitt i henholdsvis fig 6 (kun opprettet polygon) og i fig 7 (utgangspunkt og resultat).



Figur 5: Polygonet i utgangspunktet



Figur 6: Resultatpolygon



Figur 7: Utgangspunkt og resultat angitt med koordinater.

Algoritmeforslag nr 2

Dette algoritmeforslaget viderefører ideer fra det første og i en viss forstand nokså primitive forslaget. I stedet for å tillegge punktene (svært) ulik vekt i opprettingsarbeidet, så blir nå ethvert punkt justert ut fra en deling av usikkerheten mellom dette punktet og dets to nabopunkter (det foregående og det neste).

Hovedretning for resultatpolygonet velges fortsatt som den lengste sidekanten (ut fra antagelsen om at dersom usikkerheten er like stor i hvert digitaliserte punkt, så har den lengste sidekanten den minste usikkerheten når det er retning vi ser på), men nå finner algoritmen selv ut hvor lengste sidekant befinner seg i det oppgitte polygonet.

Det er fortsatt et forbedringspotensial i å analysere dataene litt mer på forhånd for å fastlegge hovedretningen og muligens i etterarbeidet ved å korrigere sidekanter som det er grunn til å tro skal sammenfalle i retning (innledningsvis sett på som en "arkitektonisk" egenskap). Begge disse har trolig liten praktisk verdi (mer estetisk muligens).

Det legges også i oppsettet av denne algoritmen at den skal være lett å gjennomskue matematisk og lite omfattende å programmere. Algoritmen bygger på følgende punkter:

- Kun hjørner skal være angitt (kun to punkt som angir en sidekant).
- Punktene er angitt i hjørnerekkefølge rundt objektet (retningen er ikke vesentlig her)
- Antall punkt (hjørner) skal være et partall.
- Usikkerheten er den samme i angivelsen av hvert punkt.
- Den lengste sidekanten angir "korrekt" hovedretning for polygonet.

Algoritmen arbeider i flere steg:

STEG 1: Gjennomsøker datasettet (punktene) og beregner lengden av hver sidekant. Finner den lengste sidekanten (dersom flere er like lange, velges den først påtrufne i nummerrekkefølge).

STEG 2: Forskyver punktenes nummerering slik at de fortsatt kommer i samme rekkefølge ved omløp av polygonet fra punkt til punkt, men det første punktet (P_1) blir nå satt til å være startpunktet på lengste sidekant (første med hensyn til den opprinnelige nummereringen av punktene).

STEG 3: Setter de to hovedretningene for det opprettede polygonet som parallell til lengste sidekant (som nå ligger først i punktlisten) og vinkelrett på denne. I regnearbeidet utgjør hovedretningene utgangspunktet for basisvektorene i dekomponeringen, som altså gir hvert av de nye punktene (faktisk forskyvning). Det blir litt mindre regnearbeid når vektorene er normert, og de omarbeides derfor til å ha lengde 1.

STEG 4: Hvert hjørne blir rettet opp. Dette skjer ved å se på fire og fire punkt om gangen og beregne nye koordinater til (forskyve) de to midterste i hver punktsekvens.

Steg 1

Alle hjørnenes koordinater (punktene) legges i rekkefølge i en tabell (eller en listestruktur). Helt standard sortering anvendes på lengdene av sidekantene. Antar at sidekanten mellom siste og første punkt i lista ($P_n \longleftrightarrow P_1$) er lengst. Sjekker denne mot de påfølgende beregnede sidelengdene til en lengre påtreffes, eller til alle er sammenlignet. Tar vare på indeksen (punktnummeret) der lengste sidekant starter.

Steg 2

Alle punktene kopieres over i en ny tabell (eller liste), der startpunktet på lengste sidekant fra steg 1 blir lagt inn først. Så kopieres punktene i samme stigende nummerrekkefølge som den opprinnelige, med avbrekk i overgangen mellom opprinnelig siste oppgitte punkt (P_n) og opprinnelig første oppgitte punkt (P_1).

Etter steg 2 ligger punktene i samme omløpsordning som opprinnelig (dvs. polygonet er selvsagt det samme, det er punktnummerne som har endret seg), men nå ligger lengste sidekant som den første sidekanten i opplistingen av polynomets koordinater.

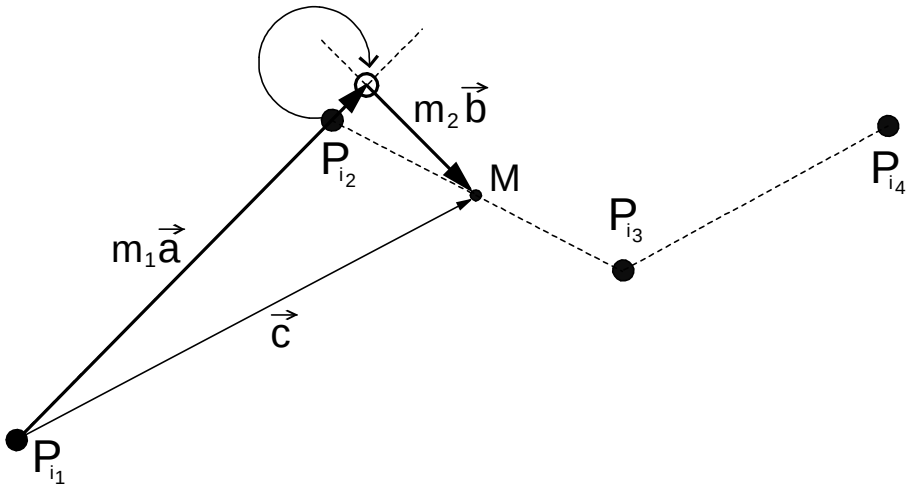
Steg 3

Hovedretningene til resultatpolygonet settes som retningen angitt ved lengste sidekant (sidekanten $P_1 \longleftrightarrow P_2$) og retningen som er vinkelrett på denne. Hovedretningene settes som basisvektorer og normeres (omregnes til lengde 1), kalt henholdsvis $\overrightarrow{a}$ og $\overrightarrow{b}$.

Steg 4

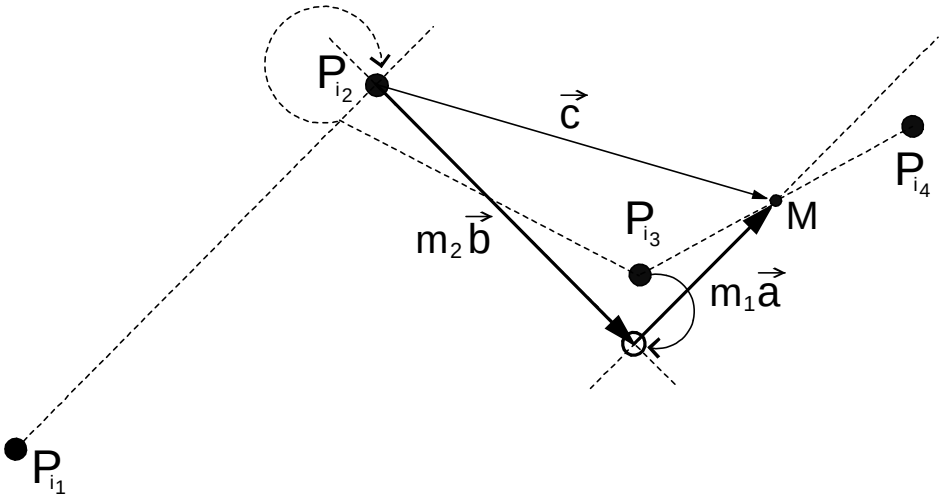
For å ha kontroll med hvilken av de to hovedretningene som er aktive fra et punkt til det neste, er algoritmen satt opp slik at den tar fire og fire punkt om gangen (fire er det minste antall punkt som kan inngå i polygonet). På fig 8 er de kalt P_{i_1} , P_{i_2} , P_{i_3} og P_{i_4} for hver gang. Indekseringen er valgt slik for å vise at det er fire punkt, og at de ligger i nummerrekkefølge. Startindeksen flyttes så to hakk frem, dvs. $i_1 = 1, 3, 5, \ldots, n-3$.

En vektor $\overrightarrow{c}$ (fig 8) strekkes fra punktet P_{i_1} til midtpunktet på sidekanten mellom de to punktene P_{i_2} og P_{i_3}. Vektoren $\overrightarrow{c}$ dekomponeres (skrives som en veid vektorsum): $\overrightarrow{c} = m_1\overrightarrow{a} + m_2\overrightarrow{b}$. Det er nå $m_1\overrightarrow{a}$ som er parallell med sidekanten mellom P_{i_1} og P_{i_2} og denne erstatter opprinnelig sidekant ved at punktet P_{i_2} flyttes til endepunktet for vektoren: $\overrightarrow{OP_{i_2}} = \overrightarrow{OP_{i_1}} + m_1\overrightarrow{a}$. Vi har altså brukt de tre første punktene i sekvensen til å forskyve P_{i_2} (O angir origo).



Figur 8 : Oppretting av første punkt-trippel

Dette regnearbeidet gjentas på de tre siste av de fire punktene (det siste punkttrippelet). Dvs. en vektor $\overrightarrow{c}$ (fig 9) strekkes fra det nye punktet P_{i_2} til midtpunktet på sidekanten mellom punktene P_{i_3} og P_{i_4} . Vektoren $\overrightarrow{c}$ dekomponeres (skrives som en vektorsum): $\overrightarrow{c} = m_1\overrightarrow{a} + m_2\overrightarrow{b}$. Nå er det $m_2\overrightarrow{b}$ som er parallell med sidekanten mellom P_{i_2} og P_{i_3} og denne erstatter opprinnelig sidekant ved at P_{i_3} får nye koordinater ved endepunktet for: $\overrightarrow{OP_{i_3}} = \overrightarrow{OP_{i_2}} + m_2\overrightarrow{b}$.



Figur 9 : Oppretting av andre punkt-trippel

Merk at indeksene i_1, i_2, i_3, i_4 kan settes fortløpende helt til $i_4 = n$ (nummeret på det siste punktet i polygonet). Siste gjennomkjøring av forskyvningen (P_n og P_1 skal på plass) krever spesiell setting av indeksene: $i_1 = n-1,\ i_2 = n,\ i_3 = 1$ og $i_4 = 2$.

Beregningene som inngår i algoritmeforslag nr 2.

Steg 1

$$L_1 = \left\|\overrightarrow{P_nP_1}\right\| = \|(p_{1x} - p_{nx}\ ,\ p_{1y} - p_{ny})\| = \sqrt{(p_{1x} - p_{nx})^2 + (p_{1y} - p_{ny})^2}$$

Gjenta for: $i = 1, 2, \ldots, n-1$

$$L_2 = \left\|\overrightarrow{P_iP_{i+1}}\right\| = \|(p_{i+1\,x} - p_{ix}\ ,\ p_{i+1\,y} - p_{iy})\| = \sqrt{(p_{i+1\,x} - p_{ix})^2 + (p_{i+1\,y} - p_{iy})^2}$$

Hvis nå $L_2 > L_1$ så noteres indeksen i og L_1 får ny verdi fra L_2 før søket fortsetter.

(Det er selvsagt nok å se på kvadratene av lengdene og kutte ut kvadratrotberegningene).

Steg 2

Dette steget inneholder kun programmering av punkforskyvningen i den opprinnelige listen. Det punktet P_i funnet i steg 1 som er startpunktet på den lengste sidekanten, blir flyttet slik at det havner først i punktlista. De øvrige punktene flyttes tilsvarende slik at punktrekkefølgen i polygonet opprettholdes. I fortsettelsen vil P_1 angi startpunktet for den lengste sidekanten.

STEG 3

Oppsetting av basisvektorene

$\overrightarrow{a} \;=\; (a_x \;,\; a_y) \;=\; \overrightarrow{P_1P_2} \;=\; (p_{2x} - p_{1x} \;,\; p_{2y} - p_{1y})$

Lengden (normen) av $\overrightarrow{a}$: $\quad \|\overrightarrow{a}\| \;=\; \sqrt{(p_{2x} - p_{1x})^2 \;+\; (p_{2y} - p_{1y})^2} \;=\; L$

Normering av $\overrightarrow{a}$: $\quad \overrightarrow{a} \;=\; (a_x \;,\; a_y) \;:=\; \dfrac{1}{L}\overrightarrow{a} \;=\; (\dfrac{a_x}{L} \;,\; \dfrac{a_y}{L})$

Vinkelrett (normalt) på $\overrightarrow{a}$, og med lengde 1 : $\quad \overrightarrow{b} \;=\; (b_x \;,\; b_y) \;= (-a_y \;,\; a_x)$

STEG 4

Behandler fire og fire punkter P_{i_1} , P_{i_2} , P_{i_3} og P_{i_4} , som deles i to punkt-trippel

I) Første punkt-trippel P_{i_1} , P_{i_2} og P_{i_3}

Midtpunktvektor: $\overrightarrow{c} \;=\; \overrightarrow{P_{i_1}M}$,

der midtpunktet M finnes som endepunktet for $\overrightarrow{OM} \;=\; \frac{1}{2}\,(\,\overrightarrow{OP_{i_2}} \;+\; \overrightarrow{OP_{i_3}}\,)$.

Dvs. $\quad \overrightarrow{c} \;=\; (c_x \;,\; c_y) \;=\; (\,\dfrac{p_{i_3\,x}}{2} \;+\; \dfrac{p_{i_2\,x}}{2} \;-\; p_{i_1\,x} \;,\; \dfrac{p_{i_3\,y}}{2} \;+\; \dfrac{p_{i_2\,y}}{2} \;-\; p_{i_1\,y}\,)$

Dekomponering: $\quad \overrightarrow{c} \;= m_1\overrightarrow{a} \;+ m_2\,\overrightarrow{b}$

Skal bare bruke parallellkomponenten $m_1\overrightarrow{a}$, og beregner derfor kun m_1.

Vektorlikningnen gir opphav til et likningssystem i to likninger med to ukjente:

$$m_1 \cdot a_x \;+ m_2 \cdot\, b_x = c_x \qquad \text{og} \qquad m_1 \cdot a_y \;+ m_2 \cdot\, b_y = c_y$$

Likningssystemets determinant er nå $a_x \cdot b_y \;-\; a_y \cdot\, b_x = 1$ som følge av normeringen av de to basisvektorene. Ved bruk av Cramers regel finner vi at: $\quad m_1 = c_x \cdot b_y \;-\; c_y \cdot\, b_x$

Forskyvning av punktet P_{i_2} til nytt: $\quad P_{i_2} = (\,p_{i_2\,x} \;,\; p_{i_2\,y}\,) = (\,p_{i_1\,x} + m_1 a_x \;,\; p_{i_1\,y} + m_1 a_y\,)$

II) Andre punkt-trippel P_{i_2} , P_{i_3} og P_{i_4} (NB! P_{i_2} er nytt forskjøvet punkt)

Midtpunktvektor: $\overrightarrow{c} \;=\; \overrightarrow{P_{i_2}M}$,

der midtpunktet M finnes som endepunktet for $\overrightarrow{OM} \;=\; \frac{1}{2}\,(\,\overrightarrow{OP_{i_3}} \;+\; \overrightarrow{OP_{i_4}}\,)$.

dvs. $\quad \overrightarrow{c} \;=\; (c_x \;,\; c_y) \;=\; (\,\dfrac{p_{i_4\,x}}{2} \;+\; \dfrac{p_{i_3\,x}}{2} \;-\; p_{i_2\,x} \;,\; \dfrac{p_{i_4\,y}}{2} \;+\; \dfrac{p_{i_3\,y}}{2} \;-\; p_{i_2\,y}\,)$

Dekomponering: $\quad \overrightarrow{c} \;= m_1\overrightarrow{a} \;+ m_2\,\overrightarrow{b}$

Skal bare bruke normalkomponenten $m_2\,\overrightarrow{b}$, og beregner derfor kun m_2.

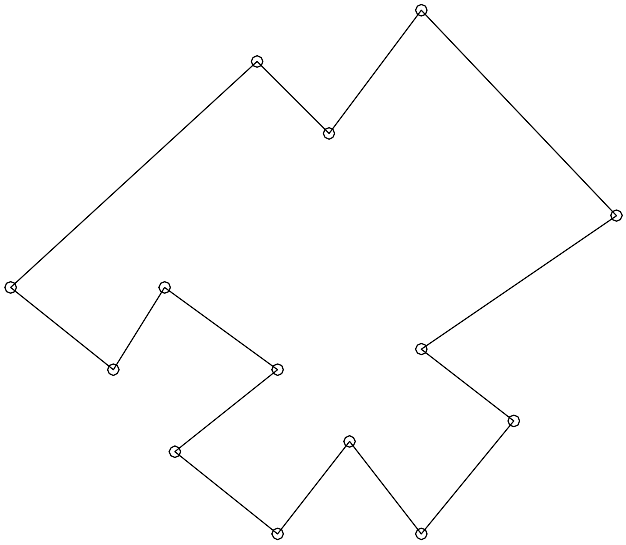
Vektorlikningnen gir opphav til et likningssystem i to likninger med to ukjente:

$$m_1 \cdot a_x \;+ m_2 \cdot\, b_x = c_x \qquad \text{og} \qquad m_1 \cdot a_y \;+ m_2 \cdot\, b_y = c_y$$

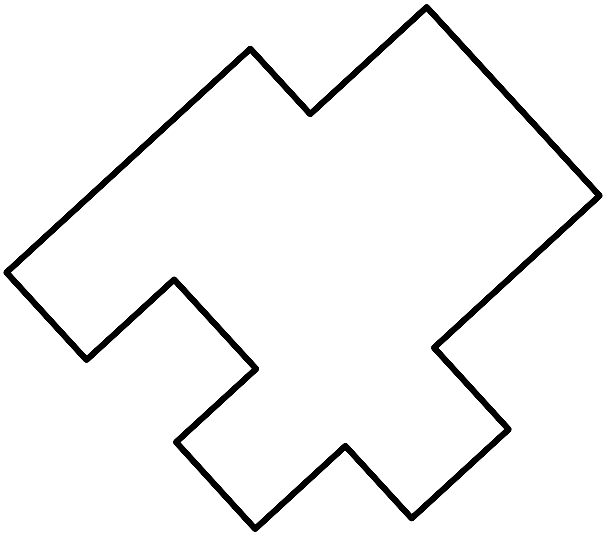
Likningssystemets determinant er nå $a_x \cdot b_y \;-\; a_y \cdot\, b_x = 1$ som følge av normeringen av de to basisvektorene. Ved bruk av Cramers regel finner vi at: $\quad m_2 = a_x \cdot c_y \;-\; a_y \cdot\, c_x$

Forskyvning av punktet P_{i_3} til nytt: $\quad P_{i_3} = (\,p_{i_3\,x} \;,\; p_{i_3\,y}\,) = (\,p_{i_2\,x} + m_2 b_x \;,\; p_{i_2\,y} + m_2 b_y\,)$

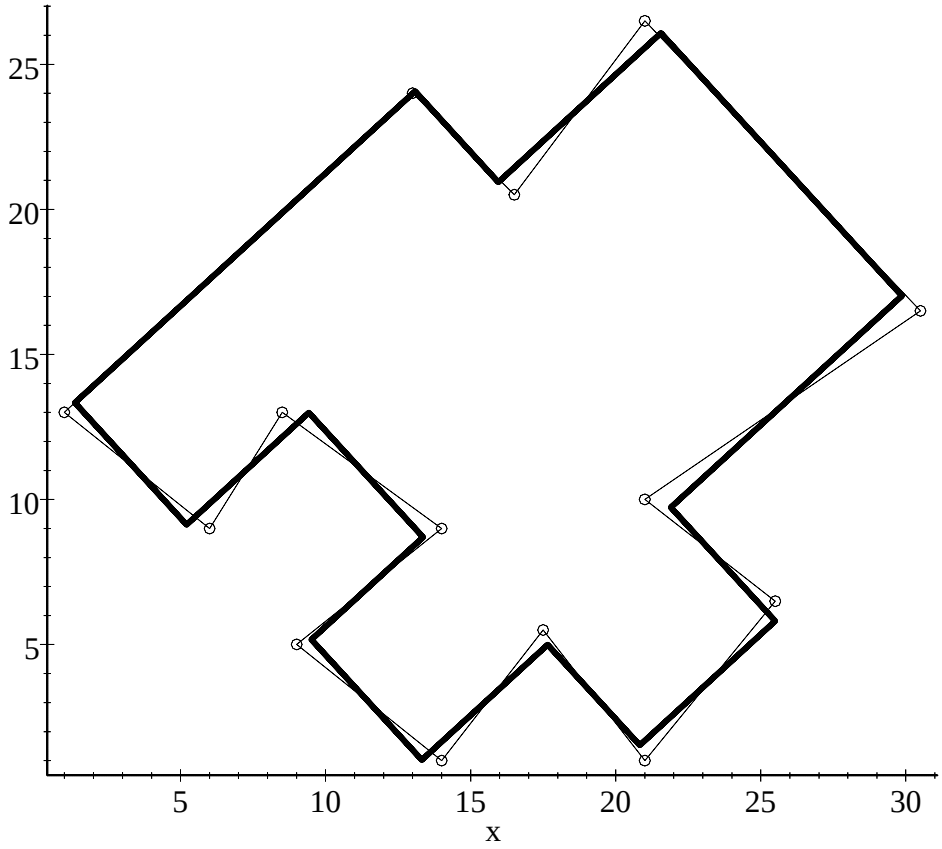
Eksempel på polygon opprettet ved algoritme nr 2.
På fig 10 har vi valgt samme polygon som representant for et digitalisert objekt som i eksemplet på bruk av algoritme nr 1. Resultatet ved bruk av andre algoritmeforslag på dette polygonet er gjengitt i henholdsvis fig 11 (kun opprettet polygon) og i fig 12 (utgangspunkt og resultat).



Figur 10: Polygonet i utgangspunktet



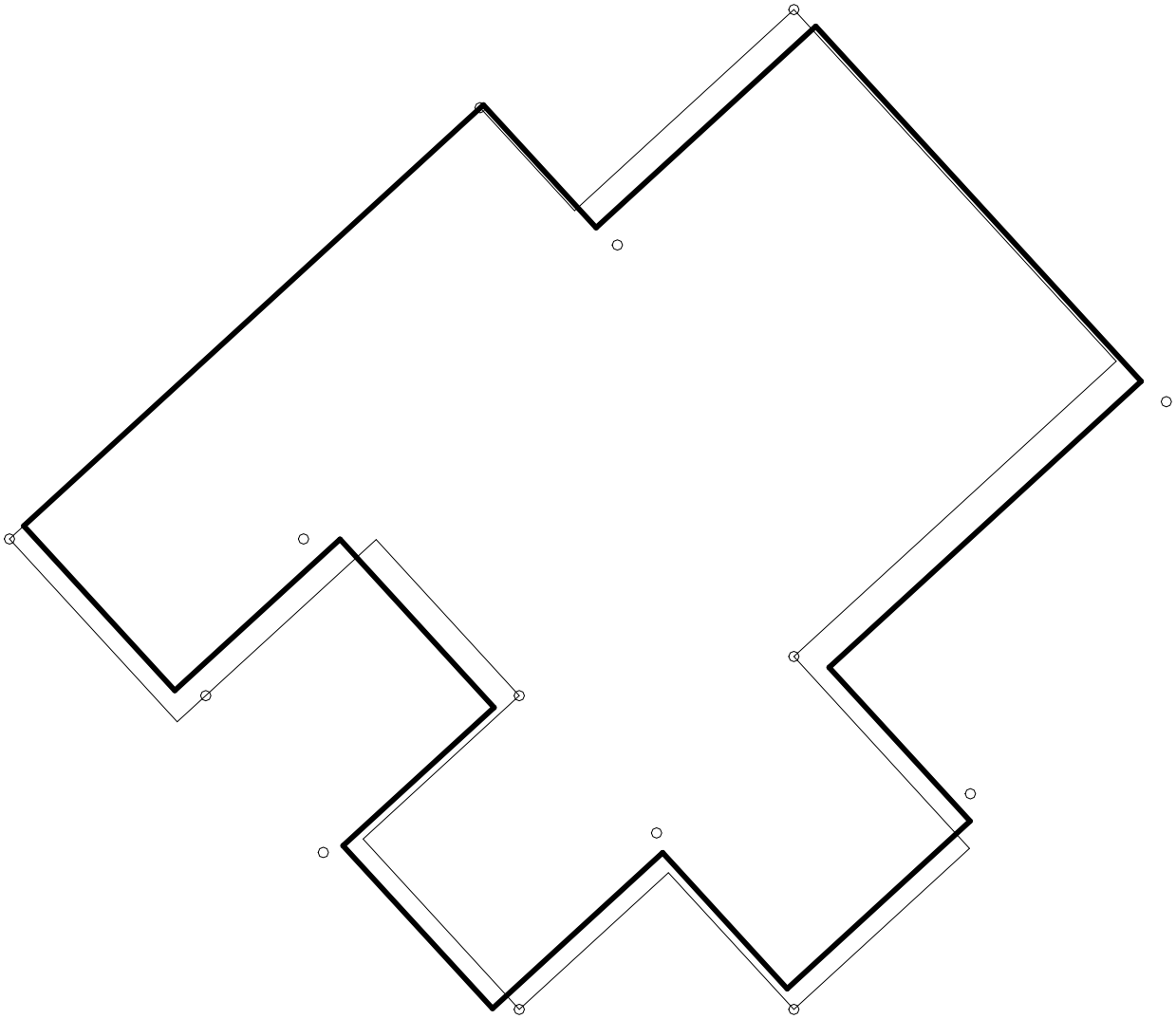
Figur 11: Resultatpolygon



Figur 12: Utgangspunkt og resultat angitt med koordinater.

Eksempel på samme polygon opprettet ved begge algoritmene

På fig 13 har vi satt opp resultatet ved bruk av både algoritme nr 1 og nr 2 sammen med punktene i det opprinnelige polygonet (det digitaliserte objektet).



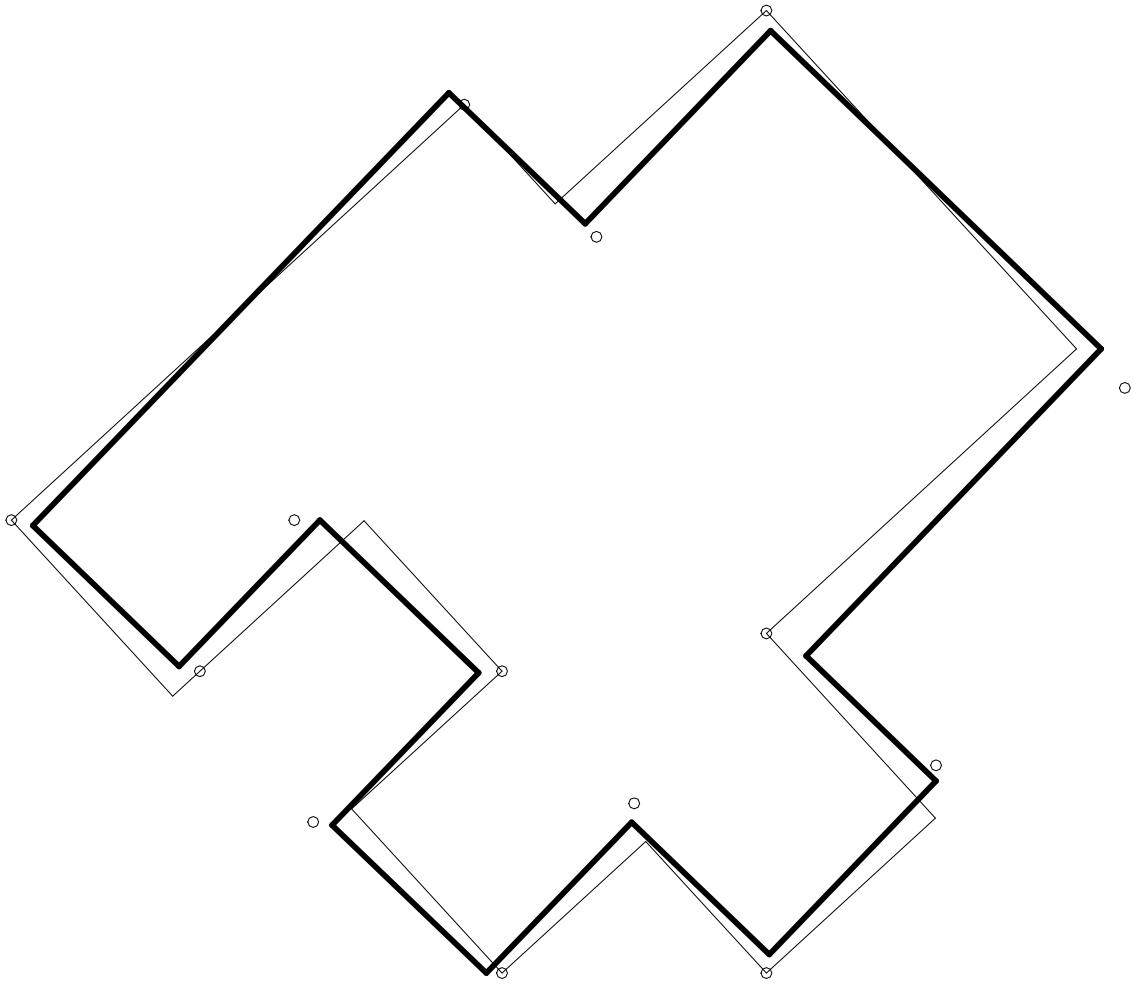
Figur 13 : Digitalisert objekt og resultatene fra to ulike opprettingsalgoritmer.

Aktuelle videreføringer

I - GEOGRAFISK ORIENTERING

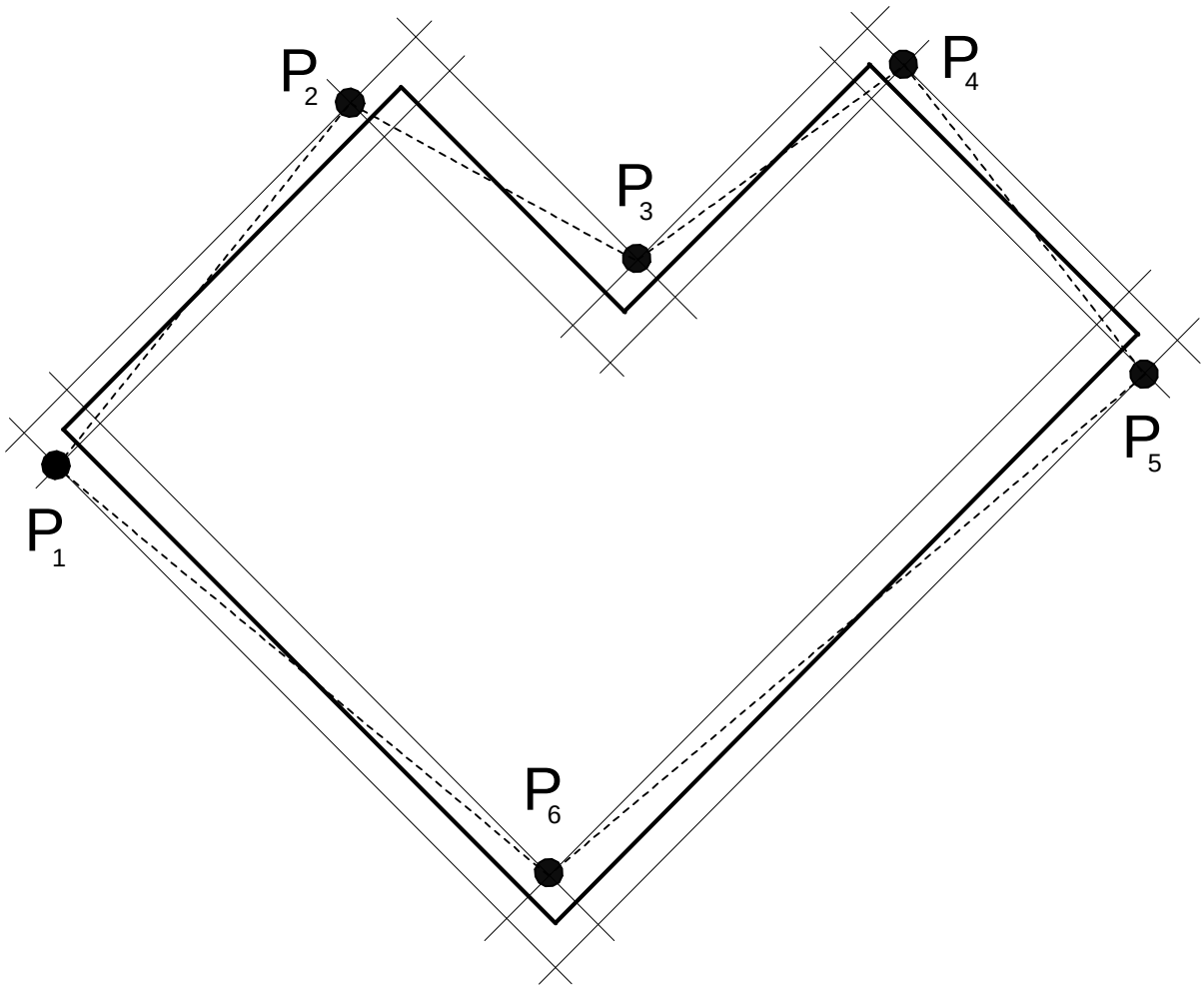
Vi kan lett ta hensyn til den geografiske orienteringen av objektet under selve valget av hovedretninger (basisvektorer) for opprettingen. I stedet for slik som i algoritmeforslagene 1 og 2 å bruke lengste sidekant og en vinkelrett på denne som hovedretninger, kan vi beregne en veid gjennomsnittsretning for alle sidekantene i forhold til en fast retning, f.eks. horisontalen eller vertikalen (dvs. koordinataksene).

Fremgangsmåten blir da å lese av vinkelen mellom hver sidekant og horisontalen (husk å enten rotere annehver sidekant 90^o, eller eventult lese av tilhørende vinkel mot vertikalen). Multipliser vinkelen med lengden av sidekanten (veier vinkelen etter hvordan usikkerheten i retningen er). Ved å dividere summen av veide vinkler med summen av sidelengder får vi veid gjennomsnittsvinkel. Denne retingen sammen med den som er vinkelrett på, lar vi utgjøre de to hovedretningene til objektet. Denne prosessen inneholder kun standard regnearbeid og er ikke tatt med her. Eksempel på polygonoppretting er gjengitt under, og programkode for hele prosessen er gjengitt i eget vedlegg.



Figur 14 : Opprettingsalgoritme som tar hensyn til geografisk orientering.

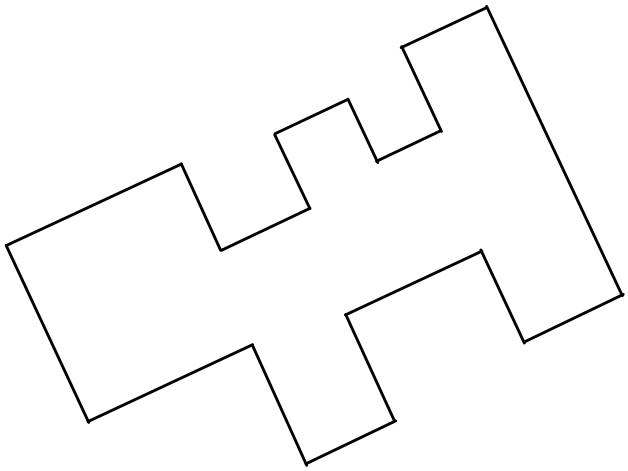
Figur 15 viser et eksempel på en polygonoppretting basert på geografisk orientering som hovedretning (ikke lengste sidekant). Figuren viser tydelig hvordan opprettingsalgoritmen beregner de nye hjørnepunktene ved en parvis halvering av usikkerhetene i de opprinnelige hjørnepunktene for hver sidekant.



Figur 15 : Hovedretning etter geografisk orientering.

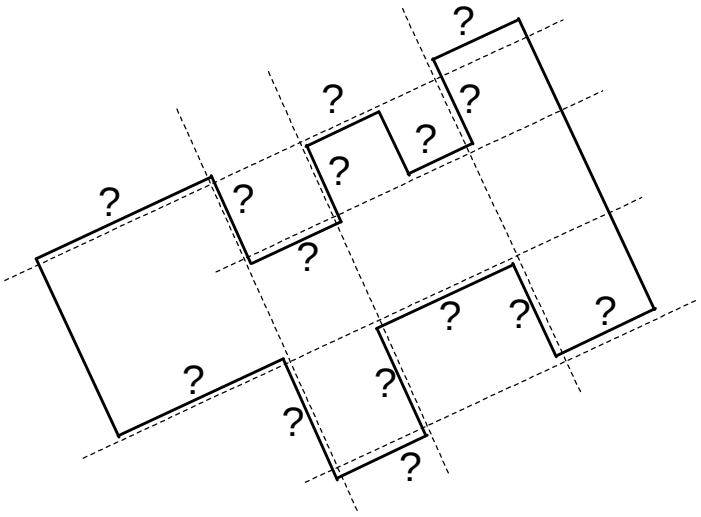
II - ARKITEKTONISK/ESTETISK ETTERJUSTERING

Resultatet av en polygonoppretting vil ved noe mere kompliserte objekt som vist ved eksempelet i fig 16, typisk reise spørsmål om noen av sidekantene bør parallellforskyves (fig 17) for å få et omriss som innfrir arkitektoniske eller estetiske krav til polygonet som representant for en faktisk bygning (fig 18). Vi har foreløpig ikke gått inn på slike justeringsalgoritmer.



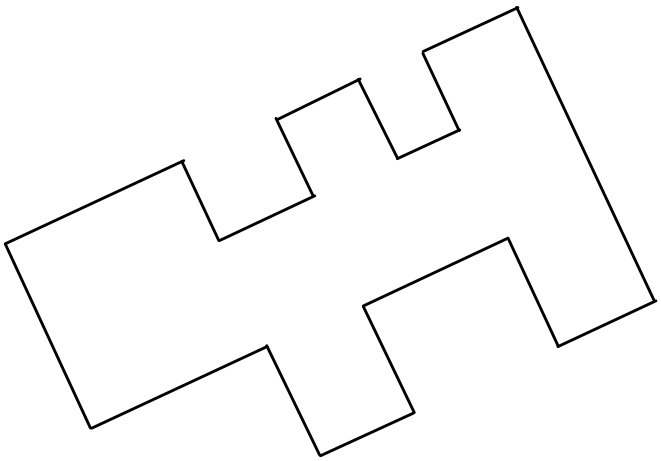
Figur 16

En av opprettingsalgoritmene har produsert et nytt polygon.



Figur 17

Bør flere av sidene parallellforskyves for å få et mer korrekt arkitektonisk utseende ?



Figur 18

Opprettet polygon etterjustert med hensyn på arkitektektonisk korrekt utseende.
Men så, burde de to åpningene på oversiden vært like brede ?

Vedlegg

Det er lagt ved et program skrevet i Pascal, for algoritmeforslag nr 2 med hovedretningene beregnet ut fra en geografisk orientering av polygonet. Dvs. at programmet beregner en veid gjennomsnittsretning for alle sidekantene som inngår i polygonet, og denne retningen brukes i arbeidet med å rette opp hjørnepunktene i polygonet ut fra prinsippet om å midle parvis usikkerhet for hjørnepunktene på hver sidekant.

```
PROGRAM Polygonoppretting(f,g,OUTPUT);
{ I denne versjonen av opprettingsalgoritmen beregnes hovedretningene i et }
{ gitt polygon ved å se på alle sidekantenes vinkel mot horisontalen.  Det }
{ antas at usikkerheten er like stor i hvert punkt, og at alle bør rettes. }
USES WinCrt;

CONST
   maxpkt = 30;   { Velger maksimalt antall hjørner for aktuelle polygoner }
VAR
                                  f, g :TEXT;
                i, i1, i2, i3, i4, n :INTEGER;
   ax, ay, bx, by, cx, cy, m1, m2, L :REAL;
    sum1x, sum1y, sum2x, sum2y, cosu :REAL;
                               px, py :ARRAY[0..maxpkt] OF REAL;
                               sL, sv :REAL;
BEGIN
   { Oppstart, åpner filer og leser inn opprinnelige punktkoordinater }

   ASSIGN(g,'c:\kladd\innpkt.txt');
   RESET(g);
   { Ytre fil for innlesing av koordinater, dvs de digitaliserte punktene }

   ASSIGN(f,'c:\kladd\utpkt.txt');
   REWRITE(f);
   { Ytre fil for utskrift av koordinater, dvs nye omregnede punkt }

   READLN(g, n);
   { Leser antall punkt, dvs hjørner i polygonet.  Krav:  n <= maxpkt }

   FOR i:=1 TO n DO
    READLN(g, px[i],py[i]);
   CLOSE(g);
   { Legger de digitaliserte punktene i to tabeller kalt px og py,
     en for x-koordinatene og en for y-koordinatene }

   {** Danner summmer av sidekantenes lengde og retning **}
   sum1x:=px[2]-px[1];
   sum1y:=py[2]-py[1];
   { Vektorsum av sidekanter som skal representere første hovedretning }

   L:=SQRT( SQR(sum1x)+SQR(sum1y) );
   { Lengden av første sidekant }

   sL:=L;
   { sL er variabel for summen av lengdene på alle sidekantene }

   sv:=L*ARCTAN(sum1y/sum1x)*180/PI;
   { sv er variabel for veid sum av alle vinkler mellom sidekant og x-akse }

   sum2x:=px[n]-px[1];
   sum2y:=py[n]-py[1];
   { Vektorsum av sidekanter som skal representere andre hovedretning }
```

```
i:=2;
REPEAT
  ax:=px[i+2]-px[i+1];  ay:=py[i+2]-py[i+1];
  cosu:=(ax*sum1x + ay*sum1y);
  IF cosu < 0 THEN
  BEGIN
   ax:=-ax;  ay:=-ay
  END;
  sum1x:=sum1x + ax; sum1y:=sum1y + ay;

  bx:=px[i+1]-px[i];  by:=py[i+1]-py[i];
  cosu:=(bx*sum2x + by*sum2y);
  IF cosu < 0 THEN
  BEGIN
   bx:=-bx;  by:=-by
  END;
  sum2x:=sum2x + bx;  sum2y:=sum2y + by;
  i:=i+2
UNTIL i = n;


{ Roterer sumvektoren sum2 over mot sum1 }
IF (sum1x*sum2y - sum1y*sum2x) < 0 THEN
BEGIN
  m1:=sum2x; sum2x:=-sum2y; sum2y:=m1
END
ELSE
BEGIN
  m1:=sum2x; sum2x:=sum2y; sum2y:=-m1
END;


{ Beregner hovedretningene a og b for oppretting av polygonet }
ax:=sum1x+sum2x;  ay:=sum1y+sum2y;
L:=SQRT( SQR(ax)+SQR(ay) );
ax:=ax/L; ay:=ay/L;
bx:=-ay; by:=ax;

{ Tar kopi av startpunkt, kopien brukes i sluttkorrigering }
px[0]:=px[1]; py[0]:=py[1];

{ Legger på plass det første hjørnet P1 }
cx:=(px[2]-px[n])/2;
cy:=(py[2]-py[n])/2;
{ Strekker vektoren c mellom midtpunktene på sidene Pn-P1 og P1-P2 }

{ Dekomponerer vektoren c i c = m1*a + m2*b }
m2:=ax*cy - ay*cx;
px[1]:=(px[n]+px[1])/2 + m2*bx;
py[1]:=(py[n]+py[1])/2 + m2*by;
```

```
  { Justerer resten av de opprinnelige punktkoordinatene }
  i:=1;
  REPEAT
    IF i+3 <= n THEN
    BEGIN
      i1:=i; i2:=i+1;
      i3:=i+2; i4:=i+3
    END  { Håndterer fire punkt i rekkefølge for hvert }
    ELSE   { gjennomløp, fastsetter to nye koordinater }
    BEGIN
      i1:=n-1; i2:=n;   { Bruker pkt P1 og P2 for å fastsette ny Pn }
      i3:=0; i4:=2      { og ny P1, derfor denne indeksoppdelingen  }
    END;

    cx:=px[i3]/2 + px[i2]/2 - px[i1];
    cy:=py[i3]/2 + py[i2]/2 - py[i1];
    { Trekker vektor c fra Pi til midtpkt på siden mellom Pi+1 og Pi+2 }

    m1:=cx*by - bx*cy;   { Dekomponerer vektoren c i c = m1*a + m2*b }

    px[i2]:=px[i1] + m1*ax;
    py[i2]:=py[i1] + m1*ay;
    { Endrer punktet Pi+1 til endepunktet for vektoren OPi + m*a }

    IF i3 = 0 THEN
     i3:=1;   { Bruker korrigert P1 istedet for opprinnelig P1 }

    cx:=px[i4]/2 + px[i3]/2 - px[i2];
    cy:=py[i4]/2 + py[i3]/2 - py[i2];
    { Trekker vektor c fra Pi+1 til midtpkt på siden mellom Pi+2 og Pi+3 }

    m2:=ax*cy - ay*cx; { Dekomponerer vektoren c i c = m1*a + m2*b }

    px[i3]:=px[i2] + m2*bx;  py[i3]:=py[i2] + m2*by;
    { Endrer punktet Pi+2 til endepunktet for vektoren OPi+1 + m2*b }

    i:=i+2
    { Tar to og to punkt siden det er to basisvektorer som selvsagt peker }
    { i to forskjellige retninger, alt er å bytte basisvektor hver gang.  }
  UNTIL i = n+1;

  { Skriver ut på en fil de beregnede punktene.  Lukker ut-fila }
  FOR i:=1 TO n DO
    WRITE(f, px[i]:1:3,',', py[i]:1:3,',');
  WRITELN(f, px[1]:1:3,',', py[1]:1:3);

  CLOSE(f);
  DoneWinCrt;
END.
```

Referanser

Howard Anton: *Elementary Linear Algebra*, 8 edt. John Wiley & Sons, 2000.

Rapporten er utarbeidet i Scientific Workplace v.3.0.

Adobe Acrobat Writer v.5.0 er brukt til å produsere elektronisk versjon av rapporten med tanke på nedlasting over internett.

Illustrasjonene på figurene 1, 2, 5, 6, 7, 10, 11, 12, 13 og 14 er alle utarbeidet i Scientific Workplace sin grafiske modul.

Illustrasjonene på figurene 3, 4, 8, 9, 15, 16, 17 og 18 er alle utarbeidet i grafikkprogrammet Mayura Draw v.3.64.

Eksemplene på polygonoppretting som er gjengitt i illustrasjonene er produsert ved simulering av de aktuelle opprettingsalgoritmene i Borland TurboPascal v.7.0.