INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2023-06-01

EXAMENSARBETE
Improvements to planar convex hull algorithms through theoretical and empirical analysis
STUDENTER Björn Magnusson, Erik Amirell Eklöf
HANDLEDARE Jonas Skeppstedt (LTH)
EXAMINATOR Michael Doggett (LTH)

Bygg en hage så snabbt som möjligt

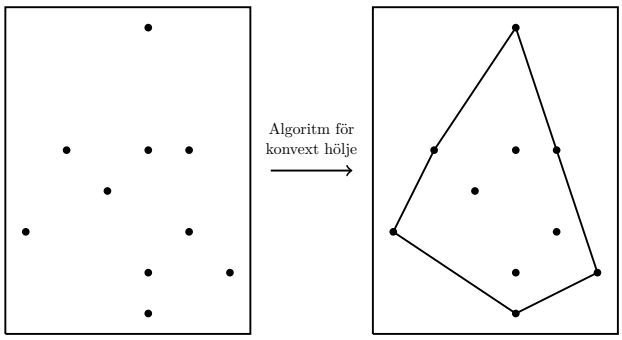
POPULÄRVETENSKAPLIG SAMMANFATTNING **Björn Magnusson, Erik Amirell Eklöf**

Att räkna ut ett konvext hölje kan användas för att jämföra hur mycket två mängder mätdata överlappar eller matcha ihop två 3D-bilder på en hjärna. I vårt arbete har vi undersökt hur man ska göra algoritmerna som räknar ut konvext hölje så snabba som möjligt.

Vi har undersökt algoritmer som räknar ut konvext hölje. Man kan tänka på det som att man vill bygga en så liten hage som möjligt runt ett antal blommor. Kanske vill man skydda dem från att bli uppätna?

Konvexa höljen kan till exempel användas för att uppskatta hur stort område en ny sjukdom har spridits till eller hur lika två grupper patienter är.

Bilden nedan illustrerar beräkningen av det konvexa höljet av ett antal punkter. I fallet med blommor skulle punkterna i den vänstra bilden representera blommor sedda ovanifrån. Dessa punkter matas sedan in i en algoritm för konvext hölje som genererar hagen i den högra bilden.



Det finns många algoritmer för att beräkna konvext hölje, som fungerar olika bra i olika sammanhang. Till exempel beroende på hur punkterna är fördelade och vilken sorts dator som används.

För att kunna räkna ut konvexa höljen snabbare och effektivare har vi undersökt vilken algoritm som blir snabbast och i vilka sammanhang. På så sätt kan konvexa höljen av stora mängder punkter beräknas snabbare och med mindre elförbrukning.

Vi har kommit fram till att algoritmen QuickHull nästan alltid är snabbast. QuickHull hittar en punkt som är på det konvexa höljet, och delar sedan problemet i två halvor med punkterna som ska vara till höger och till vänster om punkten. Detta gör även att QuickHull kan parallelliseras och bli snabbare på en dator med många kärnor.

QuickHull har dock svagheter. Om punkterna är fördelade så att det konvexa höljet måste bestå av väligt många punkter kan den ibland bli långsam. Detta kan till exempel inträffa om punkterna ligger på en cirkel. I detta fall har vi identifierat en annan algoritm som presterar bättre än QuickHull. Denna algoritm börjar med att sortera alla punkter. Oftast tar detta onödigt lång tid, men det visar sig att sorteringen lönar sig för punkter på en cirkel.

Vi har också kommit på en ny algoritm som parar ihop punkterna till större och större höljen (hagar), och ibland prövar att sätta ihop alla höljen till ett stort. Den är inte lika snabb som QuickHull, men är lätt att förstå sig på och har inte svagheter på samma sätt.