Hough transformen - ett verktyg för bildigenkänning

Jesper Andersson

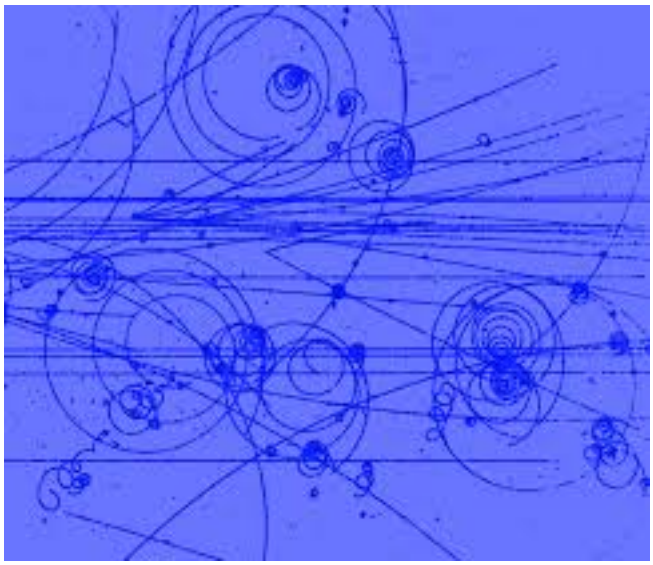
12 mars 2014

Innehåll

1 Introduktion

Säg att man har en övervakningskamera som har filmat ett område där många människor går förbi under en vecka och man vill veta om en lång person i grön hatt någon gång passerade. Man skulle kunna sätta ett gäng människor på uppdrag att gå igenom hela inspelningen, men detta skulle vara ruskigt tidskrävande. Att låta en dator gå igenom filmen och söka efter det man vill hitta och markera tidpunkter för potentiella träffar skulle kunna korta ner arbetstiden avsevärt. Den här typen av problem förekommer under området Datorseende, där man vill kunna ersätta en människas bedömning av digital information med en dators.

I den här rapporten presenteras Hough transformen, en teknik uppfunnen av Richard Duda och Peter Hart 1972, som går ut på att registrera parametriserbara matematiska objekt i digitala bilder. Ursprungligen kommer idén från Paul Hough [1], som ville underlätta processeringen av bilder från den kända bubbelkammaren som uppfanns år 1952. Bubbelkammare används för att registrera elektroners rörelsebanor och hur de påverkas av magnetfält. Genom att låta elektroner färdas genom en speciell vätska i en bubbelkammare blev spår av deras färdbanor synliga och man kunde ta bilder som visade hur de hade rört sig.



Exempel på en bild tagen från en modern bubbelkammare

Med Hough transformen ville man automatisera det tidskrävande arbetet att registrera räta linjer i dessa bilder. Man kan både få ut hur många linjer

det finns i en bild och vilken parametrisering de har. En nackdel med Hough's metod var att han använde den klassiska parametriseringen $ax + b = c$, som ger upphov till obegränsade parametrar. Richard Duda och Peter Hart kom på ett annat sätt som sätter begränsningar på parametrarna, vilket ger både snabbare beräkningshastighet och mer precision (hur mycket estimationen avviker från linjernas riktiga parametervärden). Deras metod för att registrera räta linjer är numera standard och kommer presenteras i den här artikeln.

Ett nytt användningsområde är att förenkla läkarundersökningar genom att med Hough transformen analysera blodprover och kunna bestämma antalet röda blodkroppar [5]. Anemi är ett exempel på en sjukdom som kan upptäckas genom att räkna antalet röda blodkroppar i ett stickprov. Sjukdomen gör att kroppen inte kan ta upp den nödvändiga mängden syre, ofta på grund av brist på röda blodkroppar.

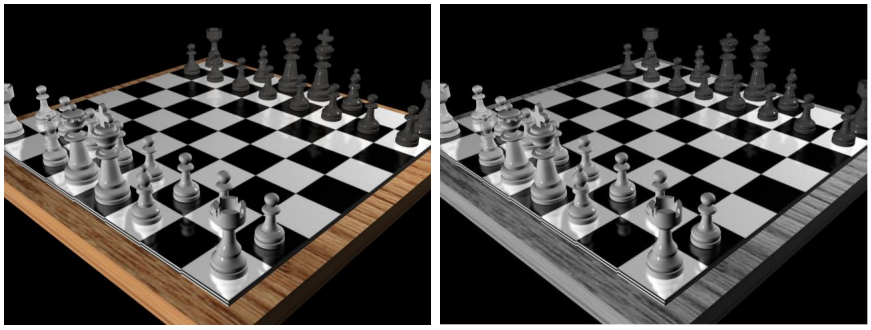
I dagsläget används två metoder för att analysera blodprover. Den ena är utförd av en läkare med hjälp av ett mikroskop. Detta är tidskrävande, och resultatet baseras på en människas individuella erfarenhet. I annat fall använder man en analysautomat, som är en mycket kostsam maskin [2]. Idén om att ersätta dessa metoder med Hough transformen kräver endast att man har en maskin som kan ta mikroskopiska bilder på blodproven och en dator som kan genomföra Hough transformen på bilderna. Man skulle kunna sätta upp en starkare dator som flera sjukhus får tillgång till via Internet, så att alla snabbt kan få sina analyser väl genomförda utan att behöva spendera stora mängder pengar på elektronisk utrustning.

För att registrera röda blodkroppar i bilder används Hough transformen för cirklar, som därför kommer att presenteras i den här rapporten. Resultatet från några exempelkörningar kommer även att visas från en egen implementation av Hough transformen för räta linjer i Matlab.

2 Teori och förberedelser

2.1 Bild och färgformat

En digital bild består av en uppsättning pixlar som har blivit tilldelade ett färgvärde. Bilderna som analyseras och behandlas kommer ha färgformatet gråskala. Det innebär att varje pixel har tilldelats ett numeriskt värde från $0, \ldots, 255$ som representerar en mjuk övergång av färger från svart till vitt. Detta värde kommer refereras som intensitet (0 och 255 är låg respektive hög intensitet). Bilder med annat färgformat kan lätt konverteras till gråskaliga bilder.



En bild med dess motsvarande gråskaliga verision

2.2 Begrepp

Ett kordinatsystem skapas över bilden med origo i mitten, skalat så att ett steg längs X-axeln är ett hopp från en pixel till dess grann-pixel.

Nu kan en gråskalig bilds intensitetsvärden betraktas som ett urval av funktionsvärden tagna från en kontinuerlig 2-d funktion $f(x, y)$ definerad över ett rektangulärt område $M \times N$ (bildens upplösning). Denna funktion kallas för bildens *intensitetfunktion*. Så varje pixel i representeras av en koordinat (x_i, y_i) och dess funktionsvärde $f(x_i, y_i)$. Med den här tolkningen av en bild kan vi göra en uppskattning av bildens intensitetsförändringar i en punkt, i form av partiella derivator. Då intensitetsfunktionen inte är given på någon analytisk form beräknas en estimation av derivatorna med hjälp av någon numerisk metod. Vi betecknar de partiella derivatorna som vanligt med $f_x(x, y)$ och $f_y(x, y)$.

En pixels bildgradient är ett mått på i vilken riktning och hur kraftigt en bilds intensitet förändras med mest styrka. Framförallt är gradientvinkeln av intresse, och den finns tillhands som en observation av de partiella

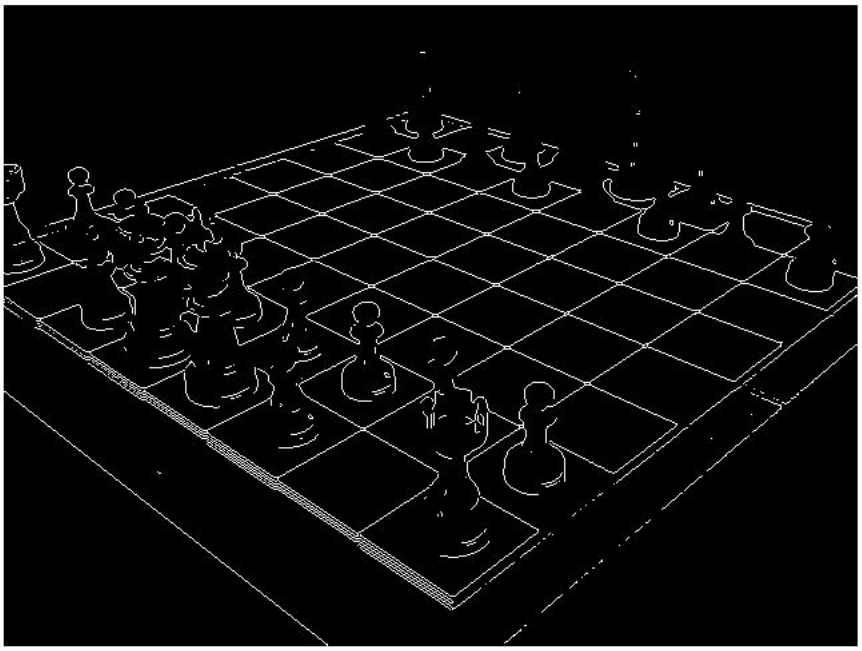
derivatorna. Bildgradient-vinkeln betecknas och definieras som

$$\theta(x, y) = \arctan \frac{f_y(x, y)}{f_x(x, y)}$$

3 Metod

Grundprinciperna för Hough Transformen där målet är att registera objekt med n paramterar fungerar så här:

- Utför en kantsegmentering på bilden. Man får då en så kallad kantbild, där varje pixel antingen har värdet 1 eller 0 beroende på om det är en pixel som tillhör en kant eller ej. Det finns utmärkta algoritmer för detta som ofta baseras på bildgradienten. Om en pixel tillhör en kant(vilket mer eller mindre innebär en stark kontrastskillnad gentemot några av dess grannar) så kommer bildgradienten få en stark magnitud och på så vis avgörs det om det är en kantpixel.

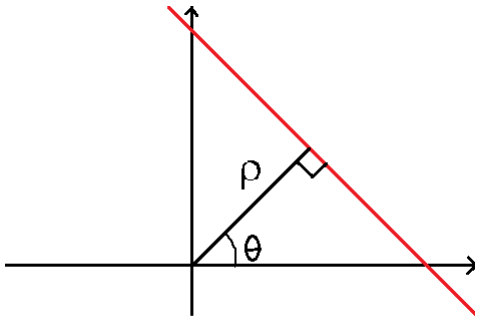


Exempel på en kantsegmentering utförd på schackbilden

- Skapa ett n-dimensionellt densitetshistogram där varje element sätts till lika med 0. Varje element i det här histogrammet har en indexering $(a_1, \ldots, a_n)$ som representerar ett potentiellt objekt i bilden med indexen som parametrar.
- Loopa igenom kantpixlar och undersök vilka parametervärden de skulle få om de tillhörde ett visst objekt. Öka histogrammets motsvarande element med 1.
- Om ett flertal pixlar faktiskt tillhör ett objekt i bilden kommer samma parametervärden ha beräknats flera gånger. Således kommer man ha ökat värdet för objektets motsvarande elementet i histogrammet med många 1:or. Objekt med högt värde i arrayen anses som "hittade".

3.1 Räta linjer

Linjer parametriseras av deras normalform (ρ, θ) där ρ är längden på normalvektorn från origo till linjen, och θ är vinkeln mellan X-axeln och normalvektorn. Man tillåter negativa ρ, därmed begränsas vinkeln mellan $0 \leq \theta < \pi$ för entydig representation.



En linjes normalform

En dator kan dock inte hantera intervall med reella tal, så en diskretisering måste göras. Man introducerar en konstant $\theta_{step} = \frac{\pi}{k}$ som svarar mot den minsta skillnaden mellan två vinklar som kan användas. Intervallet görs då om till en mängd av k vinklar som har potential till att associeras med linjer i bilden, $\{0, \ldots, \pi - \theta_{step}\}$. Ett mindre θ_{step} ger alltså fler vinklar som kan användas och därmed mer precision, men det kommer medföra att linjedetekteringen blir mer tidskrävande. I och med att man bara intresserar sig för linjer som faktiskt visar sig i bilden är även ρ begränsad, då linjer med

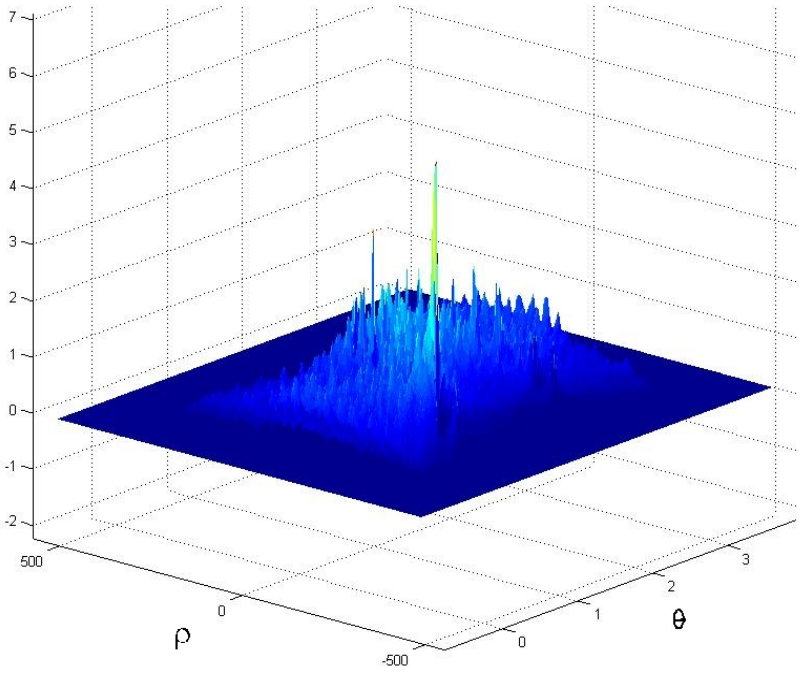
$\rho > \sqrt{\frac{M^2}{4} + \frac{N^2}{4}}$ (avståndet från bildens mittpunkt till något av dess hörn) inte kommer skära bildområdet. Låt $\sqrt{\frac{M^2}{4} + \frac{N^2}{4}}$ betecknas med ρ_{max}. Även här krävs det att man diskretiserar intervallet $[-\rho_{max}, \rho_{max}]$ genom att välja något l så att möjliga värden på ρ blir $\{-\rho_{max}, -\rho_{max} + \frac{2\rho_{max}}{l}, \ldots, \rho_{max}\}$.

Nu kan man börja gå igenom kant-pixlarna och beräkna potentiella $(\rho, \theta$-par. För varje pixel (x_i, y_i) beräknas ρ för alla möjliga θ med formeln

$$\rho = x_i \cos\theta + y_i \sin\theta$$

avrundat till närmaste tal i listan av möjliga ρ. Så för varje kant-pixel får man en mängd med par $\{(\theta_1, \rho_1), \ldots, (\theta_k, \rho_k))$. Notera att om några pixlar faktiskt ligger på en rät linje i bilden så kommer paret (θ_v, ρ_v) som motsvarar linjen de ligger på förekomma i alla dessa pixlars par-mängder.

I fallet med räta linjer får densitetshistogrammet dimmensionen $n = 2$ och storleken $l \times k$. Här svarar alltså element med index (i, j) mot en linje med parametrar θ_i och ρ_j, från parameterlistorna definierade ovan. Notera att varje element har fått värdet 0 från början. Man vill nu använda mängden av parameter-par som beräknades tidigare. För varje par (θ_i, ρ_j) så ökas värdet på element (i, j) i arrayen.



Densitetshistogram för schackbilden

Nu kommer man till momentet då man låter datorn leta efter lokala extremvärden i densitetshistogrammet. Att göra den här sökning rakt av utan någon extra information om vad man letar efter kommer sällan ge önskat resultat. Vid användning av Hough Transformen är det vanligt att man kan ge datorn lite mer att jobba med, beroende på vad man vill få ut. Exempel kan vara att man söker precis 8 linjer, eller kanske att man vill hitta så många parallella linjer som möjligt. Man får helt enkelt anpassa koden lite till sitt ändamål.

Proceduren då man fyller histogrammet med värden baserat på vad man räknade ut tidigare kan göras på olika sätt. En relativt ny teknik för Hough Transformen är att använda statistiska kernels till att skapa en densitetsestimation av linjeförekomsten i bilden [3] . Den här tekniken har fördelarna att man kan behöva färre kantpunkter och att man får ett mjukare histogram än den tidigare beskrivna metoden. I en bild då man till exempel har en ganska tjock linje så kommer det troligen i den vanliga metoden att bildas två toppar i histogrammet där linjerna de representerar tillsammans bildar den tjocka linjen, medans för kernelmetoden så får man ett klart och tydligt extremvärde för den lite tjockare linjen.

Ett alternativ till proceduren då man för varje pixel går igenom all möjliga vinklar är att man bara använder pixelns bildgradient-vinkel. Då det ofta är starka kontrastskillnader vid linjer så är det naturligt att en pixels gradient är rätvinklig mot linjen och på så vis kan det funka bra som en estimering av linjens θ-värde. Den här metoden är mycket snabbare, men mindre precis. Det är inget som garanterar att en pixels gradient är rätvinklig mot linjen, så man får räkna med lite felmarginaler. Det är då passande att använda kernel-metoden för estimeringen av histogrammet. Man får då toppar som inte är speciellt spettsiga och tydliga eftersom pixlar på samma linje kan få vinklar som med små skillnader mellan varandra, men i vissa sammanhang kan det vara användbart när man prioriterar beräkningshastighet över precision.

3.2 Cirklar

Hough transformen för cirklar brukar oftast förutsätta att man vet radien hos cirklarna som man vill hitta, det är fallet som presenteras här. Det är fullt möjligt att söka igenom en bild efter cirklar med obestämd radie, men det kräver väldigt mycket mer beräkningskraft. Här presenteras metoden då man har en fixerad radie, som betecknas r. Vad som återstår då är att hitta cirklarnas mittpunkt i form av koordinater som vi kallar (a, b).

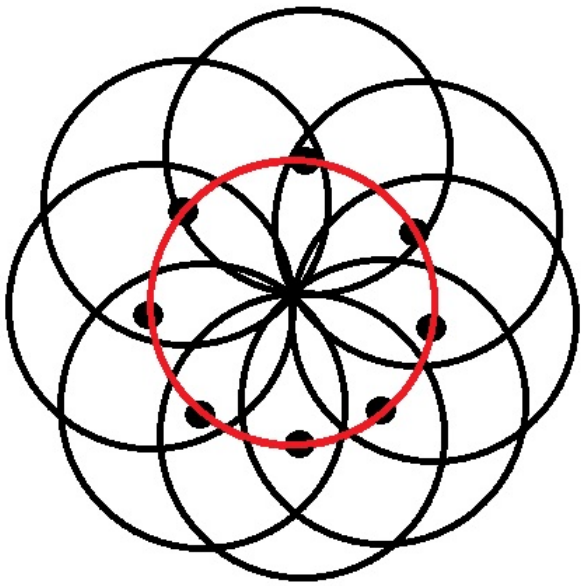
Precis som för räta linjer skapas ett 2-dimensionellt densitetshistogram

där varje punkt motsvarar en möjlig mittpunkt för en cirkel i bilden. Man utför även en kantsegmentering som tidigare beskrevs. Nu vill man beräkna potentiella mittpunkter. Om en pixel i ligger på randen till en cirkel kommer dess kordinater (x_i, y_i) uppfylla

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

där (a, b) är cirkelns mittpunkt. Man gör en mängd med olika värden på a likt den för linjer som innehöll möjliga värden på θ. a kommer vara begränsat av $x_i + r$ och $x_i - r$, annars kan inte pixeln ligga på randen. Då kan man iterativt gå igenom olika värden på a mellan dessa gränser, och räkna ut motsvarande b genom $b = y_i \pm \sqrt{r^2 - (x_i - a)^2}$. Allt eftersom ökar man densitetshistogrammets motsvarande punkter för de olika paren (a, b) som man får fram. För pixlar som faktiskt ligger på en cirkel i bilden kommer det då beräknas en viss mittpunkt (a, b) flera gånger, som kommer bli synbart i histogrammet. En sökning efter lokala maximan kan nu göras och vart eventuella cirklar befinner sig bestämms.

I bilden nedan vill den röda cirkeln med given radie r registreras.



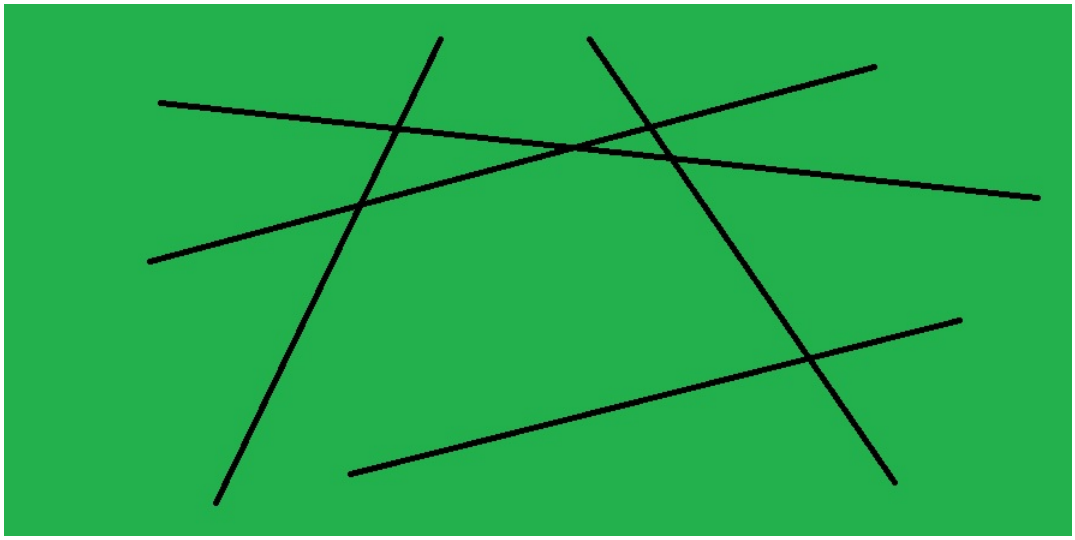
Ett antal kantpixlar(de svarta punkterna på den röda cirkeln) har valts ut. Notera att varje sådan punkt har en svart cirkel runt sig, även den med radie

r. Den här svarta cirkeln visar alla möjliga mittpunkter som den röda cirkeln skulle kunna ha, givet att den svarta pixeln ligger på dess rand. Alla dessa svarta cirklar korsar varandra i den röda cirkelns riktiga mittpunkt, allltså kommer den beräknas många gånger och ge ett högt värde i densitetshistogrammet.

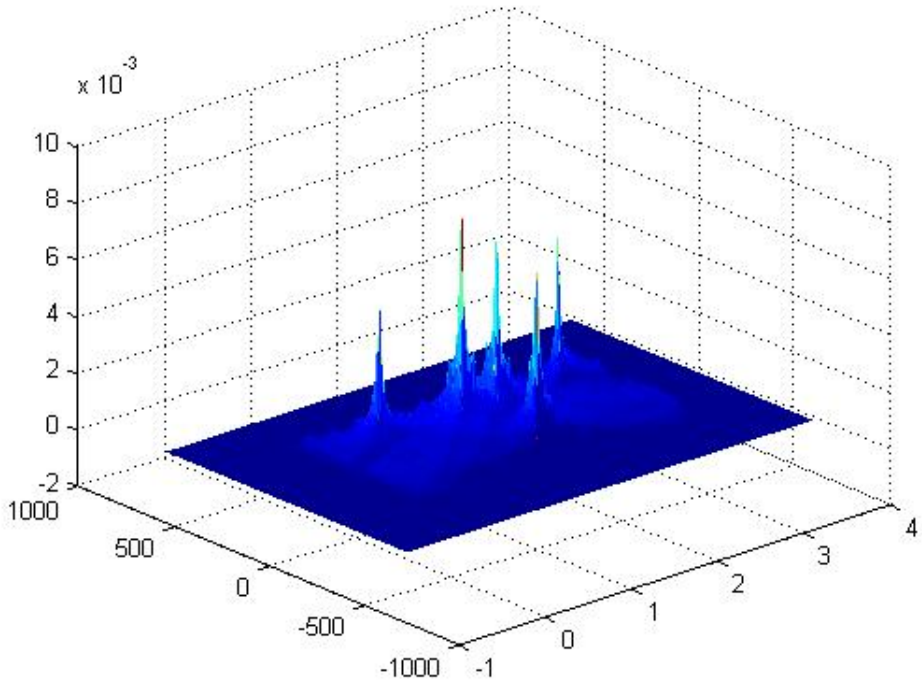
4 Metodanvändning

Här demonstreras några exemplen på vad Hough transform kan åstadkomma. I mitt program som utför Hough transformen anger man antalet linjer man vill registrera, så genererar den ett densitetshistogram för bilden som man har angivit. Programmet väljer sedan ut de högsta topparna i histogrammet och de korresponderande linjerna ritas ut i originalbilden.

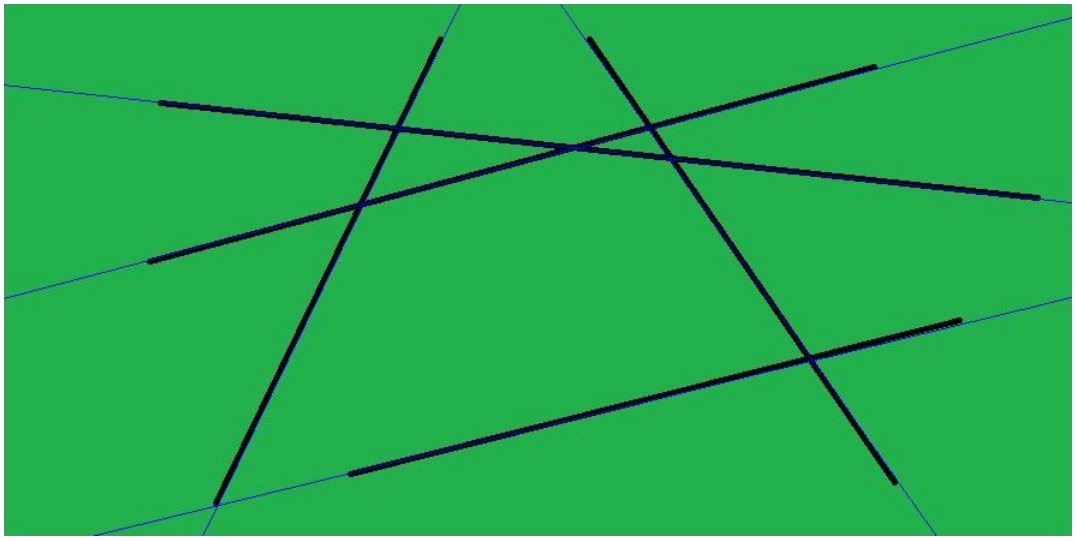
Först presenteras en mycket enkel bild med 5 räta linjer, utan några övriga objekt i bilden som kan tänkas påverka genererandet av densitetshistogrammet. I exemplet sätts θ_{step} till $\frac{\pi}{360}$ för att få en bra noggranhet på linjerna som registreras.



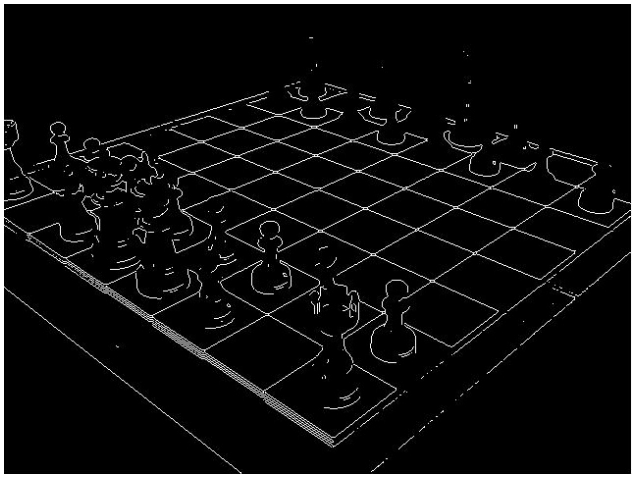
Den kantsegmenterade bilden blir i princip identisk med originalbilden, då de enda färgskillnaderna sker vid linjerna. Så alla svarta pixlar i bilden blir kantpixlar, medan resten blir icke-kantpixlar. Densitetshistogrammet blir som förväntat mycket tydligt, då de enda koordinaterna i histogrammet som får ett högt värde är de som faktiskt motsvarar linjerna i bilden.



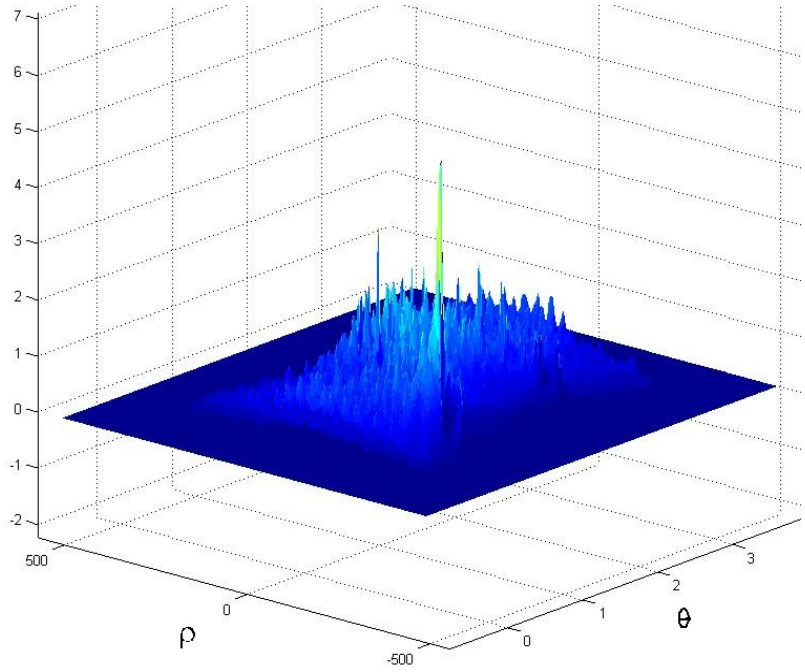
5 spetsiga toppar tar form. Topparnas koordinater i densitetshistogrammet är parametervärdena på linjerna som ritas ut i blått på originalbilden nedan.



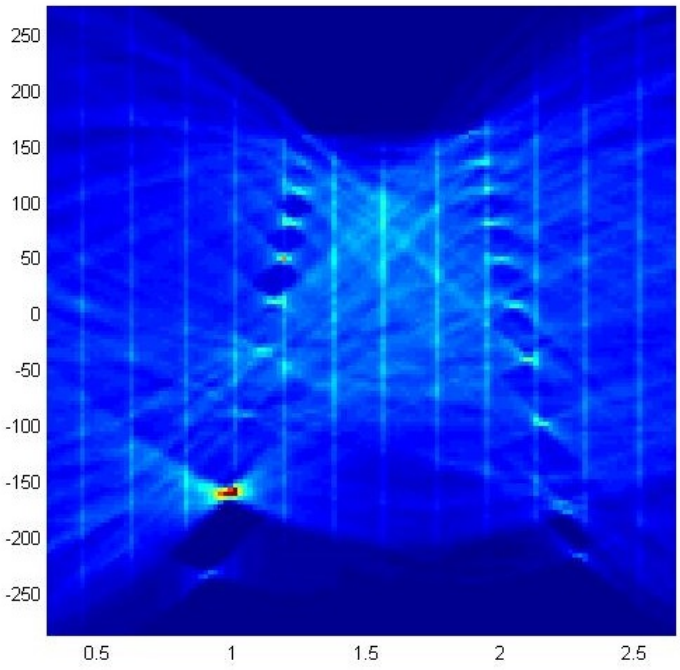
Schackbilden som har använts tidigare i rapporten är lite mer komplicerad. I den kantsegmenterade bilden ser man att det är många kantpixlar som inte tillhör någon rät linje, till exempel de pixlar som utgör kanterna till schackpjäserna. Dock är det viktigt att komma ihåg att givet en pixel, vilken som helst, så kommer det för varje annan pixel beräknas ett par (ρ, θ) som representerar linjen som går igenom dessa två pixlar. Poängen med Hough transformen är dock som tidigare påvisat att för pixlar som faktiskt ligger på en linje kommer det specifika par (ρ, θ) som representerar linjen att beräknas många gånger.



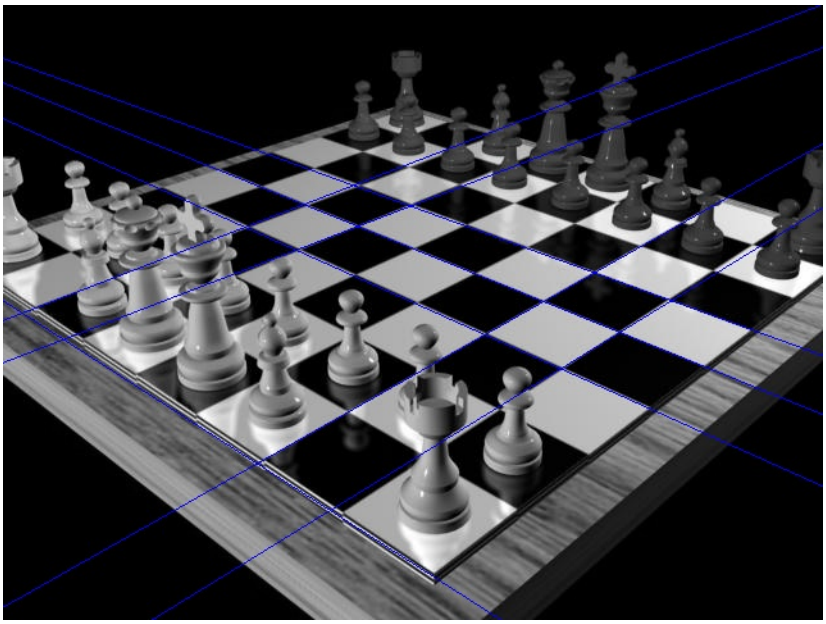
Programmet ställs in på att registrera 8 linjer. Densitetshistogrammet genereras. Det blir mer kompakt med många toppar på grund av de kantpixlar som inte tillhör någon linje. En topp blev dock väldigt mycket högre än de andra. Den motsvarar den lite tjockare linjen som går längs schackpjäserna närmast kameran. En tjock linje innebär många pixlar som kommer att generera i princip samma parametervärden, och ge ett stort värde i densitetshistogrammet.



Tittar man på histogrammet ovanifrån blir det lite tydligare att det faktiskt är några toppar som sticker ut lite jämfört med de övriga. Med vyn ovanifrån tolkar man starka färgvärden som hög densitet. De två raderna av toppar bildas av de vertikala och de horisontella linjerna i schackbrädet. Vinklarna på linjerna varierar lite, på grund av vinkeln som bilden är tagen ifrån.



8 av topparna i histogrammet väljs ut och linjerna återskapas i originalbilden. Resultatet blir bra, trots det röriga densitetshistogrammet.



Referenser

[1] Hart, P., 2009. “"How the Hough Transform was invented”, IEEE signal processing magazine.

[2] Hälso- och sjukvårdsförvaltningen, 2012. Region Gotland, “Delårsrapport 1”.

[3] Rozen, D., 2009. “Statistical Hough Transform”, IEEE Transactions on pattern analysis and machine intelligence, VOL 31, NO.8.

[4] Sonka, M., Hlavac, V. och Boyle, R., 2007. “Image Processing, Analysis and Computer Vision”, 3:e upplagan, CL engineering.

[5] Venkatalakshmi B. och Thilagavathi K., 2013. “Automatic Red Blood Cell Counting Using Hough Transform”. Department of TIFAC-CORE in Pervasive Computing Technologies, Velammal Engineering College, Chennai, India, Proceedings of IEEE Conference on Information and Communication Technologies.