Mönsterigenkänning och följning

Digitala projekt VT 2004

Carl Loodberg, d00cl@efd.lth.se, grupp 10

Abstract

The goal of this project is to construct a system capable of localizing a designated pattern visually and to keep it in sight. A standard USB camera is used for image capture and the motor to rotate the camera is a stepper motor from an old disk drive.

The software consists of three major sections, image capture and buffering, image analysis and camera movement. C# was used as programming language because it makes it easy to interface with the webcam.

Although the image analysis was not implemented to run on the Graphics Processing Unit, GPU, as initially intended the analysis runs reasonably well on the CPU. About four to five images per second can be analyzed which is enough to track a slow moving target.

Innehåll

Inledning

Målet med mitt projekt är att konstruera ett system som kan lokalisera förutbestämda mönster visuellt i sin omgivning och följa dem med blicken.

Att göra kameran rörlig åstadkoms genom att montera en webbkamera på en stegmotor. Bilden från webbkameran överförs på vanligt sätt till datorn via USB. Motorn styrs med hjälp av parallellporten och motorn försörjs med spänning från datorns nätaggregat.

För att känna igen mönster i bilden fångas bilden från webbkameran och lagras i en buffert. Bilder från bufferten analyseras sedan i den takt som datorn klarar av och det mål som söks lokaliseras.

Mjukvaran för att hämta upp bilden från webbkameran samt att styra motorn skrivs i C# och exekveras på CPU:n i datorn. Då Houghtransformen är mycket beräkningskrävande var målet att utföra den på grafikkortets GPU, Graphics Processing Unit.

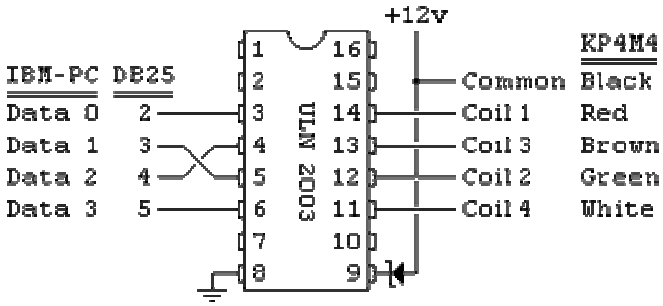
Projektet skiljer sig till stor del från traditionella projekt i kursen ”Digitala Projekt” men det inkluderar likväl en ren hårdvarudel genom styrningen av stegmotorn. Uppgiften kan delas in i tre övergripande delar nämligen att styra stegmotorn, att hämta upp och buffra bilder från webbkameran samt att analysera de upphämtade bilderna för att lokalisera målet.

Hårdvara

Motor

Den motor som använts är en stegmotor som tagits från en kasserad 5 ¼ tums diskettstation. Den matas med 12 V och har en steglängd på 3,6 grader. Genom att lägga spänning på två intilliggande lindningar samtidigt kan dock steg på 1,8 grader uppnås. Det hade gått lika bra att använda till exempel en servomotor i stället men eftersom stegmotorn var fanns tillgänglig valde jag att använda den.

Motorn styrs med hjälp av fem ledare från parallellporten, jord + en till varje lindning. För att ge tillräcklig spänning och ström till motorn används en darlingtondrivare. 12V spänning och jord tas ifrån nätaggregatet i datorn.



Figur 1. Beskrivning av hur en darlingtondrivare kopplats in mellan parallellporten och motorn för att ge tillräcklig spänning till motorn.

Kamera

Den kamera som används är en Avermedia Intercam Elite som kan ta bilder med upplösning upp till 640x480. För att analysen av bilder ska gå tillräckligt snabbt används i praktiken upplösningarna 320x240 och 160x120 vilket är tillräckligt för att kunna hitta det sökta målet.

Det kontrollerchip som sitter i kameran är OV511 från OmniVision. De DirectX-funktioner som används gör dock att kameran kan bytas ut mot valfri webbkamera som fungerar i Windows utan att mjukvaran behöver modifieras.

Färdig hårdvara

Motorn och styrelektroniken har byggts in i en liten låda. De kablar som behövs till kameran är:

- USB till webbkameran.
- Styrning av kameran från parallellporten.
- Strömförsörjning 12 V.

Eftersom man inte har någon kontroll över huruvida datorn skickar ut ettor eller nollor på pinnarna innan styrprogrammet startas kan strömförsörjningen brytas så att inte alla spolar matas på en gång då det resulterar i överhettning av motorn.

Figur 2. Den färdiga hårdvaran.

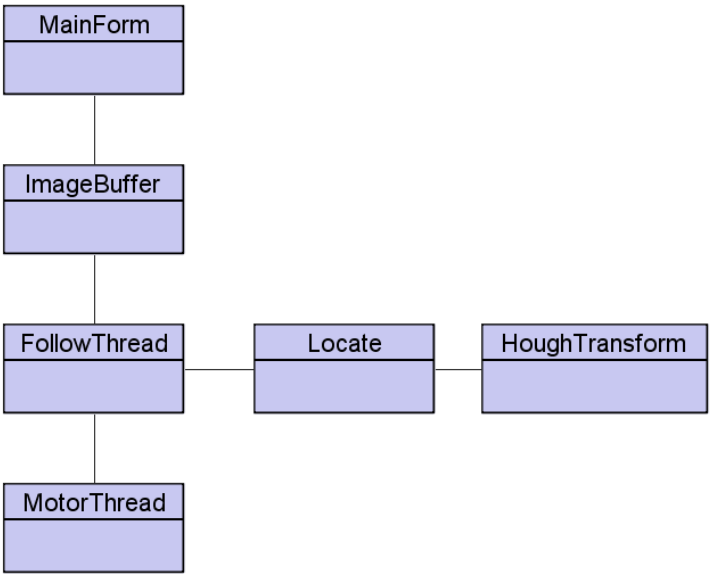
Programdesign

Språket C# som har stora likheter med både Java och C++ har använts för utvecklingen av mjukvaran. Anledningen till att jag valt detta programmeringsspråk är att det är relativt lätt att komma åt DirectX och därmed webbkameran.

De uppgifter som skall utföras är:

- **Bildhämtning**. Bilder hämtas upp från kameran och lagras i en buffert.
- **Bildanalys**. Utför bildanalys på de bilder som lagrats i bufferten i så hög takt som möjligt så att målet kan detekteras.
- **Motorstyrning**. Styr motorn till rätt riktning.

För var och en av dessa huvuduppgifter skapas en egen tråd.



Bildhämtning

Bilden fångas från webbkameran via Microsofts DirectX. Det gör att bilder kan hämtas från en godtycklig källa som stöder Windows standardinterface för videoinfångning vilket gäller de flesta webbkameror, tv-kort etc. Den klass som tar hand om detta är MainForm som även hanterar användargränssnittet i applikationen.

Den senaste bilden lagras i bufferten ImageBuffer som skyddas av en monitor för att se till så att data inte hämtas och lagras precis samtidigt. I bufferten lagras också information om hur lång tid den senaste bilden funnits i bufferten och därmed kan ett värde på hur snabbt bildanalysen utförs fås fram.

Bildanalys

Mönstret som skall kännas igen är ett antal koncentriska cirklar likt en darttavla. Orsaken till detta val är att man, med hjälp av att utföra en Houghtransform på bilden, kan lokalisera centrum hos cirklar med en viss diameter. Eftersom diametern i bilden blir olika stor beroende på

avståndet till målet så är det fördelaktigt om målet utgörs av måna koncentriska cirklar så att någon cirkel alltid bli lagom stor för att detekteras.

Teori

För att hitta de koncentriska cirklarna appliceras först sobeloperatorn på bilden för att hitta övergångar mellan ljusa och mörka områden, det vill säga kanter. Det innebär att bilden faltas med två olika matriser, en i x-led och en i y-led.

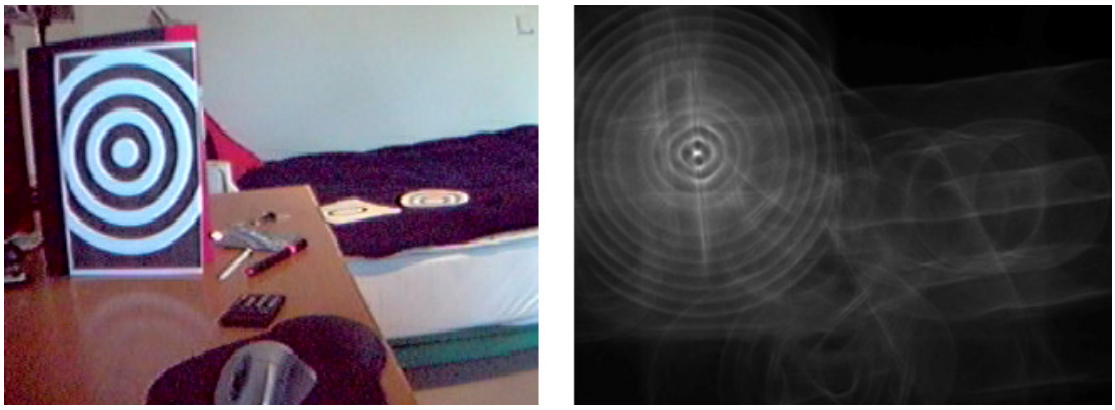
-1	0	+1
-2	0	+2
-1	0	+1

x

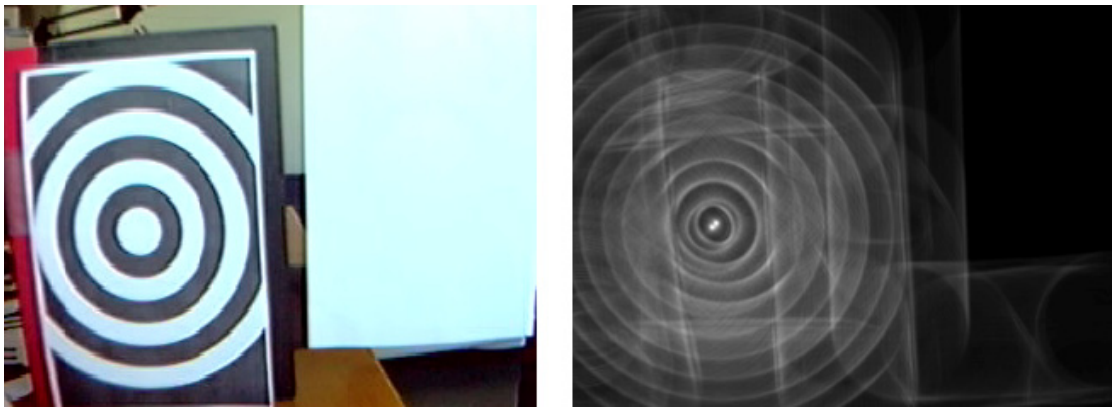
+1	+2	+1
0	0	0
-1	-2	-1

y

När kanterna detekterats i bilden transformeras bilden med cirkulär Houghtransform vilket gör att cirklar med en viss bestämd diameter kommer att ge upphov till ljusmaxima vid cirkelns centrum i den transformerade bilden. För att få reda på vart kameran skall vridas kontrolleras var ljusmaximum i x-led ligger.



Figur 3. Exempel 1 på originalbild och transformerad bild.



Figur 4. Exempel 2 på originalbild och transformerad bild.

Implementering

Bildanalysen utförs på processorn i datorn vilket att går relativt långsamt eftersom processorn inte är byggd för bildanalys. Konsekvensen av detta blir att det eftersökta mönstret ej kan röra sig för snabbt eftersom mönstret då missas. Vid en upplösning på

160x120 bildpunkter kan bilden dock analyseras 4 – 5 gånger per sekund på en Athlon XP 1800+. Målet var att flytta över dessa tunga beräkningar till grafikkortet men tiden räckte inte till.

Motorstyrning

Tråden MotorThread håller reda på kameras nuvarande riktning och den målriktning som kameran styrs mot. För att styra kameran anropas en metod som ändrar målriktningen i förhållande till den nuvarande riktningen. Med jämna mellanrum uppdateras vilka lindningar som skall slås på för att få motorn att rotera. Man skulle kunna tänka sig att en ojämn och oförutsägbar belastning på processorn skulle göra motorns rörelser ryckiga eftersom Windows XP inte är något realtidsoperativsystem men så är inte fallet.

Möjliga förbättringar

Motorstyrning

Styrningen av motorn skulle kunna förändras så att en Atmel ATmega16 tar emot kommandon via ett seriellt interface från datorns serieport och i sin tur styr motorn. Ett exempel på ett API som skulle kunna användas för detta ändamål finns på http://motion.sourceforge.net/tracking/iface-api.php.

Bildanalys på GPU

En stor förbättring skulle vara att flytta bildanalysen från processorn till grafikkortet eftersom moderna grafikkort har programmerbara ”pixel-shaders” som lämpar sig väl för detta ändamål. De grafikkort som kan komma i fråga för detta är DirectX 9-kort med stöd för något av språken assembler, HLSL (High Level Shading Language) eller Cg (C for graphics). Jag har använt ett ATI Radeon 9800 Pro som stöder HLSL i mina tester.

Pixel-shaders är från början till för att skapa realistiska ytor vid realtidsrendering i till exempel spel. Genom att lägga in den bild som skall analyseras som en textur vilket kan liknas vid en tapet, i grafikkortet kan avancerad bildbehandling utföras mycket snabbt.

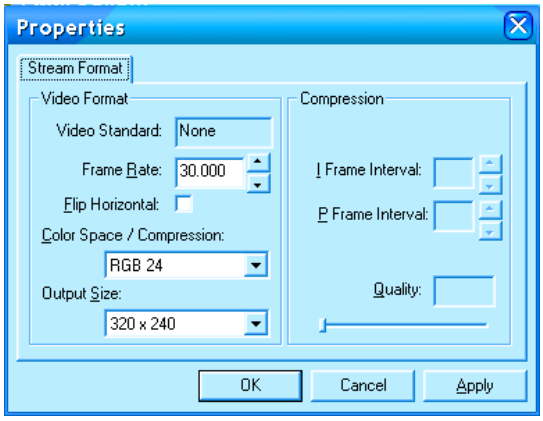
Figur 5. Exempel på kantdetektering utförd på grafikkortet.

Jag har gjort försök med att flytta beräkningar till GPU:n. Till exempel har kantdetekteringen implementerats utan problem. Att överföra utdata från GPU:n till CPU:n är mera problematiskt eftersom GPU:n är byggd för att ge utdata till skärmen

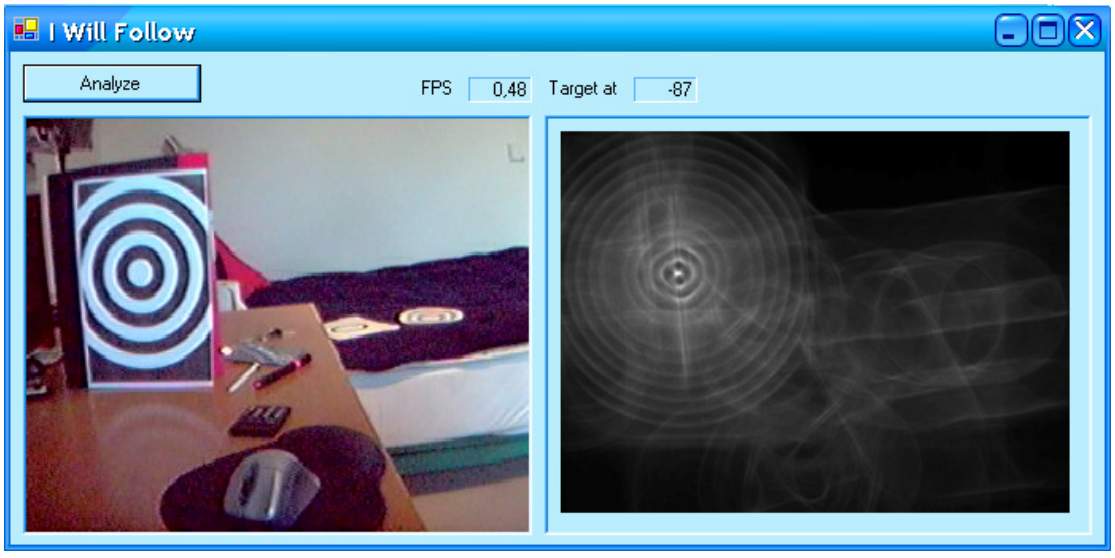
och inte till CPU:n. Problemet bör gå att lösa genom att skriva resultatet till en textur som sedan kan läsas av datorn.

Användargränssnitt

Vid uppstart väljer man vilken kamera som skall an vändas samt vilken upplösning de infångade bilderna ska ha. Även den hastighet som datorn hämtar bilder från kameran kan ställas in. Analysen av bilder sker dock maximalt i den hastighet som datorn klarar av.



Användargränssnittet innehåller en knapp för att starta följningen av målet. Två fält berättar hur många bilder per sekund som analyseras respektive hur långt från centrum som målet hittats i x-led.



Resultat

Projektet har gett erfarenhet av att kombinera enkel hemmabyggd hårdvara, avancerad off-the-shelf hårdvara och mjukvara. Det har gett mersmak för att göra avancerade beräkningar på grafikkortets GPU och visat att det finns stor potential den typen av tillämpning. Jag kommer att fortsätta arbetet med att flytta över bildbehandlingsoperationerna från CPU:n till GPU:n efter projektets slut.

Källförteckning

- M. Atiquzzaman, A Coarse to Fine Search Technique to Detect Circles In Images
  http://citeseer.ist.psu.edu/139353.html

- M. A. Schulze, Circular Hough Transform Demonstration
  http://www.markschulze.net/java/hough/index.html

- B. Fisher et al, Feature Detectors - Sobel Edge Detector
  http://www.cee.hw.ac.uk/hipr/html/sobel.html