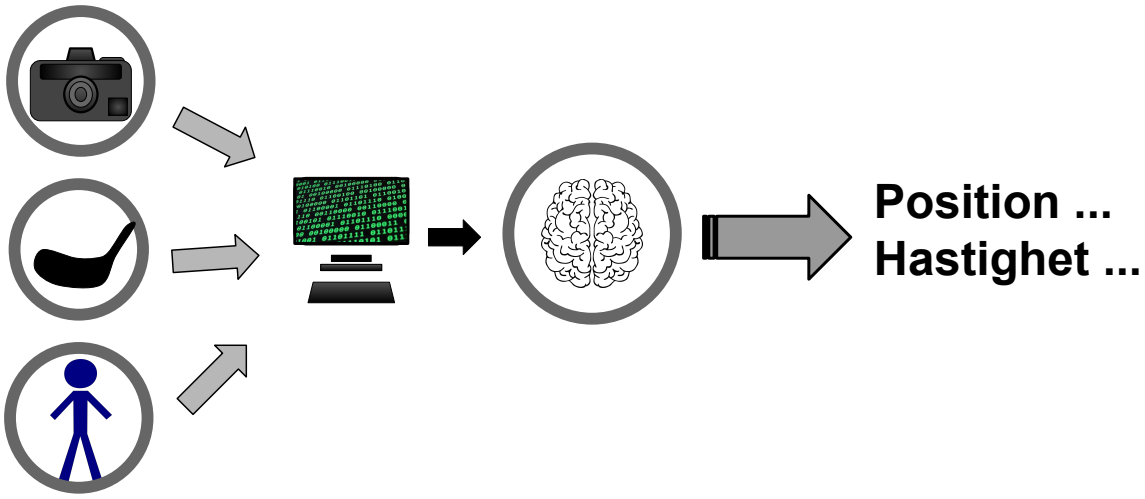
GÖTEBORGS UNIVERSITET

INSTITUTIONEN FÖR FYSIK

ANNA



Kevin Vu
Amina Warsame
Nicklas Lindevall

Uppsats för avläggande av naturvetenskaplig kandidatexamen med huvudområdet fysik
2020, 180 hp
Grundnivå

Kandidatuppsats 2020

ANNA

Ai av Neurala Nätverk för Analys av innebandyspelare

Kevin Vu
Amina Warsame
Nicklas Lindevall

Institutionen för fysik
Göteborgs Universitet
Göteborg, Sverige 2020

Handledare: Jonathan Weidow & Magnus Karlsteen
Examinator: Martina Ahlberg

Kandidatuppsats 2020
Institutionen för fysik
Göteborgs Universitet
SE-412 96 Göteborg
Telefon +46 31 772 1000

Omslag: Illustration av konceptet bakom ANNA

Gothenburg, Sweden 2020

Utveckling och mätning av ett realtids-spårningssystem till innebandy
Nicklas Lindevall, Kevin Vu, Amina Warsame
Instutitionen för fysik
Göteborgs Universitet

Sammanfattning

Inom detta kandidatarbetet har ett positioneringssystem utvecklats för att bestämma innebandyspelares hastighet samt position i realtid. Med hjälp av AI och kamera har vi uppmätt en persons hastighet som springer på en innebandyplan. Vi har tränat en AI detektor på COCO datasetet att bara känna igen personer och sätta unika IDn. Positionen för innebandyspelare detekterades och projicerades med hjälp av homografi på en bild över en innebandyrink. Genomsnittshastigheten uppmättes med maximalt relativt fel på 15 % gentemot den verkliga medelhastigheten och det genomsnittliga relativa felet på 5,14 %. Vi uppmätte den maximala fluktuationen (av hastigheten) hos en människa som försöker springa med konstant hastighet till medelhastigheten ±24 %. Minst 82 % av hastighetsmätningarna var inom ±24 % av den verkliga medelhastigheten. Detta projekt är genomfört med mål att visa det är möjligt att utveckla ett realtids-spårningssystem för att höja mervärdet av innebandy som publiksport eller till att användas som träningsredskap för klubbar. Vi kallar vårt system för ANNA (Ai av Neurala Nätverk för Analys av innebandyspelare).

Abstract

Within this bachelor thesis a positioning system has been developed to determine position and speed of floorball players in real-time. With the use of AI and a camera we have measured players speed running on a floorball field. We have trained an AI detector on the COCO dataset to recognize people and place uniqe IDs on them. Position for the floorball players are detected and projected with the help of homography on an image of a floorball rink. The average velocity was measured with a maximum relative error of 15% compared to the actual average velocity and the average relative error was 5,14%. We meatured the maximal fluctuation (of the velocity) of a person attempting to run at constant speed to be the average velocity ±24%. At least 82% of the velocity meaturements were within ±24% of the actual average velocity. This project was performed with the goal to show that it is possible to develop a real-time tracking system of increase the value to floorball as an entertainment sport or to be used as a training tool for clubs. We have named our system ANNA (Ai of Neural Networks for Analysis of floorball players).

Nyckelord: Innebandy, CNN, AI

Förord

Vi vill tacka våra handledare Magnus Karlsteen och Jonathan Weidow för vägledning genom projektet.

Nicklas Lindevall, Kevin Vu, Amina Warsame, Göteborg, Juni 2020

Innehåll

1

Introduktion

1.1 Bakgrund

Idag finns det positioneringsystem som låter åskådare från olika idrottsarrangemang följa spelares rörelsemönster i realtid. Ett av de mest avancerade och nyaste systemen är Intels True View [1]. True View använder sig av 38 stycken små 5k kameror runt en arena som mäter höjd, bredd och djup. All denna data överförs till ett serverrum i arenan och sedan till ett team av personer som analyserar datan. Med hjälp av True View kan real-tidsdata (i stort sett) mätas för att ge publiken ett mervärde till sporten. System som True View är förhållandesvis dyra för en mindre idrottsklubb och en orimlig investering. Därför finns det behov av att hitta billigare och mer användarvänliga system som även mindre idrottsföreningar kan införskaffa sig [2].

En problemlösningsmetod inom fysiken är programmering och användning av neurala nätverk. Ofta måste man behandla en stor mängd data och utföra beräkningar som inte är möjliga för hand. Detta gör algoritmer och mjukvara till en självklar metod. Därav är många av dessa algoritmer en råvara för alla typer av utvecklare, inte minst för kommersiella aktörer. För att komma i kontakt med dessa typer av mjukvara räcker det med att ta en titt på din smarta telefon vilket mest troligt har en fingeravtrycksläsare eller ett ansiktsigenkänningssystem. Dessa ansiktsigenkänningssystem använder sig med artificiell intelligens för att känna igen personer.

Objekt-detektion kan utföras med hjälp av datorseende och artificiell intelligens. Ett generellt krav för att kunna konstruera en bra detektor är träning och ett väldigt stort dataset. I avsnittet teori förklarar vi mer i detalj vad dataset är och hur man använder ett sådant för att träna en artificiell intelligens.

I detta arbete kommer vi behandla vårt egna positioneringsystem ANNA, med hjälp av ANNA kan vi mäta hastighet och position inom ett begränsat område. ANNA är en prototyp som visar att det går att utveckla enklare och billigare alternativ till de kommersiella positioneringsystemen.

1.2 Syfte och problemformulering

Syftet med kandidatarbetet är att utveckla ett positioneringssystem så att innebandyspelares hastighet och position kan bestämmas i realtid. För att göra detta används en kamera i samband med ett faltningsnätverk. Idealt ska positioneringssystemet fungera på en 20 x 40 meter innebandyplan. Arbetet är ett startprojekt för att lägga en grund till framtida liknande projekt hos sport och teknologi, Chalmers.

1.3 Avgränsningar

Detta kandidatatarbete var tidsbegränsat då projektet genomfördes som ett halvtidsarbete under en termin. Projektet är även begränsat då det bestod av tre medlemmar med samma utbildningsbakgrund (individuella arbetsinsatser kan ses i appendix). Detta påverkade våra resultat, dock inte i den mån att det inte är tillräckligt för att dra slutsatser.

En förundersökning till projektet genomfördes för att utvärdera vilket tillvägagångssätt som skulle användas för att utveckla positioneringssystemet. Utifrån undersökningen drog vi slutsatsen att det bästa alternativet var att använda sig av kameror och artificiell intelligens istället för de övriga alternativen, sensorer och radar. Slutsatsen drogs att det blir svårt att behandla alla spelare som individuella punkter när de kommer alldeles för nära radarn, för övrigt är radardetektorer bra när det kommer till material med elektrisk ledningsförmåga då dessa reflekterar signaler bättre (alltså inte människor). Tröghetssensorer var ett annat alternativ som avfärdades. Dessa sensorer (accelerometrar och gyroskop) har förmåga att mäta bärarens avstånd och hastighet i realtid. Detta är ett billigare alternativ men kan dock påverka spelarens prestation och ge missvisande data. Den missvisande datan beror på vart vi väljer att placera sensorerna. Exempelvis kan handrörelser eller fotrörelser tolkas som gång eller sprint, vi antog att filtrering skulle bli ett för stort problem. Fysiska sensorer som sitter på spelare löper även risken att skadas vid kollision med andra spelare eller marken.

Vi har valt att använda en kamera för livesändning och utveckla en artificiell intelligens för att analysera bilden och spåra spelares position samt hastighet.

1.4 Etiska ställningstaganden

Vid konstruktion av ett artificiellt neuralt nätverk tillkommer även ansvar. Det man huvudsakligen gör är att bygga en självtänkande enhet som tar dess egna beslut samt har möjligheten att förbättra sig själv. Detta skapar en olägenhet för människan och en riskzon där oönskad information har en potentiell förmåga att läcka ut eller missanvändas på något sätt. Det är viktigt att man sätter säkerheten före målet. Inom vårt projekt sparas inga personliga detaljer av spelarna så som ansikte eller namn. Spelaren är tänkt att bara identifieras av klassen människa och lagnummer, vilket är de enda relevanta informationen systemet behöver. Vidare distribueras inte informationen som systemet uppfattar till någon annan part.

Relationen mellan teknik och samhället är inte en enkelriktad gata utan deras växelverkan sker i båda riktningarna. Detta innebär att där det finns en efterfrågan och ett behov tillkommer även ett utbud. Konceptet bakom utvecklingen av ett positioneringsystem är inte något ovanligt. Som nämnts tidigare finns det idag kommericiella positioneringsystem som marknadsförs till klubbar. Anledningen till varför dessa företag blir framgångsrika är för att det existerar ett behov i marknaden. Emellertid är dessa system inte prisvärda för mindre klubbar. Därav finns det behov av billigare system som lever upp till samma prestation om inte bättre. Målet med detta arbetet är att utforska möjligheterna för att skapa ett sådant system med begränsade resurser.

Idealt kan en mindre klubb följa spelares statistik och hitta förbättringsområden med vårt system. Detta kan leda till högre prestation för laget som en helhet. En konsekvens med detta är att systemet kan utnyttjas i fel syften. Där man jämför spelare mot varandra, vilket kan leda till påfrestningar i lagarbetet. En påföljd blir att en eller flera gruppmedlemmar blir syndabockar för gruppen. Detta kan skapa mental ohälsa hos spelarna och därmed leda till ohälsosamma vanor. För att handskas med detta på bästa sätt är det viktigt att tränaren tar kontroll över situationen och förklarar syftet med detta system. Syftet är inte att en enskild spelare ska förbättras utan laget som en helhet ska optimera deras spel. Frågan om det är etiskt och moraliskt rätt att skapa en sådan detektor kan diskuteras. Då det uppenbarligen existerar både för-och nackdelar i frågan. På ena sidan har man en ojämn spelplan där små klubbar inte har råd med dagens kommersiella system och går miste om förbättringsmöjligheter. På andra sidan är vi medvetna att systemet kan utnyttjas för fel syften.

Att inte utforska möjligheten för en billig och anpassningsbar detektor verkar orimlig. Då man genom att inte påbörja en undersökning går miste om en möjlighet att hjälpa sina lokala klubbar. Ett positioneringsystem gör det inte bara roligt för spelarna att följa deras utveckling, utan det sätter även perspektiv på saker som annars hade gått obemärkt förbi. Det vi väsentligen gör är att skapa förutsättningar för mindre klubbar. Eftersom detektorn inte sparar någon personlig information är det inte skadligt. ANNA är begränsad på så sätt och inkräktar därmed inte mot människans integritet och på denna grund betraktar vi inte vårt projekt som

omoraliskt.

2
Teori

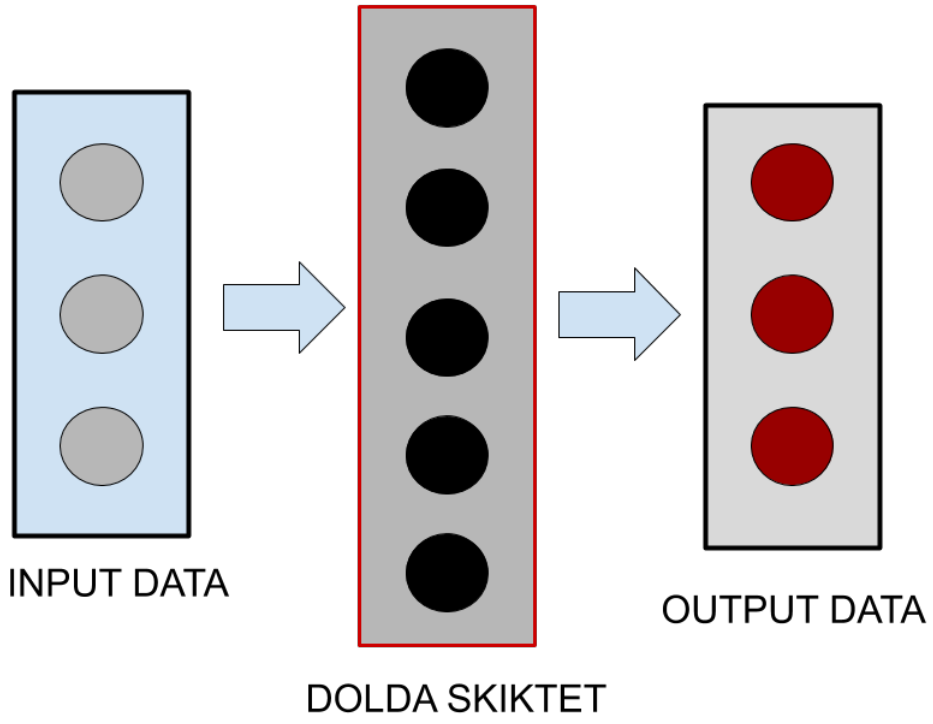
2.1 Datorseende

Datorseende (Eng. computer vision) är vetenskapen om att manipulera eller förståelse av bilder och videoklipp. Idag används datorseende som tillämpning i mängder av olika fält från självkörande bilar till medicinsk diagnostik.

Inom datorseende kan man detektera objekt och klassificera dem till olika kategorier, det vill säga skilja på spelare och boll exempelvis. I stort sätt är konceptet med datorseende att ge datorn ögon. Till skillnad från det mänskliga ögat kan algoritmen få ett semantiskt gap, vilket innebär att översättningen mellan högnivåspråket och maskinspråket blir allt sämre. Detta kan leda till att datorn avläser pixlarna men inte kan avgöra vad den ser. Detta är ett problem som framträder vid bildanalys, dock finns det flera sätt att åtgärda detta problem varav ett av dem är maskininlärning.

2.2 Artificiella neurala nätverk (ANN)

Artificiella neurala nätverk jämförs nästan alltid med hjärnan och dess arbetssätt. Då själva konceptet påminner om det verkliga nervsystemet där våra hjärnor utnyttjar en enorm mängd kopplade neuroner för att bearbeta all information. Neuronerna i ett artificiellt nätverk är arrangerade i olika skikt där det första skicktet består av ett inputskikt och därefter har man dolda skikt där inputdatan genomgår en mängd operationer. Dessa olika steg leder vidare till det sista skiktet, outputskiktet där resultatet kan observeras, se figur 2.1.



Figur 2.1: De tre olika generella stegen i en ANN. Första steget visas till vänster där inputdatan läses in och därefter bearbetas i det dolda skiktet. I slutändan hamnar den bearbetade datan i outputskiktet som genererar och visar resultatet.

2.3 Faltningsnätverk

Engelskans convolutional neural network (CNN), även kallad för faltningsnätverk på svenska, är en typ av ANN som är speciellt användbart för bildigenkänning. CNN karakteriseras av sitt sätt att hitta objekt på bilder genom att successivt läsa igenom hela bilden i små steg. Därmed analyserar inte algoritmen bilden i sin helhet i en enda operation utan i olika sekvenser. På detta sätt skapas en matematisk sammanställning av de olika sekvenserna som därefter anger hur mycket de efterliknar det eftersökta objekten [3].

Grunden för varje faltningsnätverk består i stort sätt av följande operationer; faltning, ReLU (Rectified Linear Unit), pooling och slutligen klassificering [4].
Denna sortens arkitektur för djupinlärning var först introducerad av Kunihiko Fukushima 1980 i neocognitron [5].

I det tidigare stycket har vi nämnt att CNN karakteriseras med sättet den avläser bilder på med en sekvens i taget. Det är detta som menas med faltning och därav även namnet faltningsnätverk. En bild klassificeras som en matris med pixelvärden. Vad faltningsnätverket då gör är att använda ett så kallat filter i formen av en matris som utför en elementvis matrismultiplikation med inputbildens matris, vilket resulterar i en ny matris som kallas för en funktionskarta (Eng. feature map) [4]. Filtret består av vikter och väljes efter vilket resultat som är av intresse. Därefter utförs en kompletterande operation på datan som kallas för Rectified Linear Unit (ReLU). Syftet med denna operation är att öka ickelinjariteten i bilderna för att förtydliga övergången mellan olika pixlar, färger och gränserna för olika objekt.

För att minska på beräkningutrymme utnyttjar man pooling, vilket i stort sätt är en funktion som minskar den rumsliga representationen för den resulterade bilden. Pooling sker i samband med varje faltning och ReLU operation i de olika skikten i det neurala nätverket [6]. På detta sätt minskar dimensionen på matrisen samtidigt som den avgörande informationen bibehålls för att senare kunna leda till klassificering.

Klassificering är den slutliga processen i det dolda skiktet som innebär att nätverket klassar objekt som den har identifierat. Klassificeringen sker i det sista lagret av nätverket, när alla faltningslager är anslutna till varandra för att gemensamt bidra till en sannolikhetsfördelning för olika klasser [4]. Det som observeras efter det dolda skiktet är outputdatan som presenterar denna sannolikhetsfördelning.

2.3.1 Dataset

Dataset innebär enligt Sveriges riksdag en samling av strukturerad data som kan laddas ner för att senare bearbetas [7]. I vårt arbete har vi inkluderat två olika dataset: COCO som står för *common objects in context* samt SVHN som står för *street view house numbers* [8].

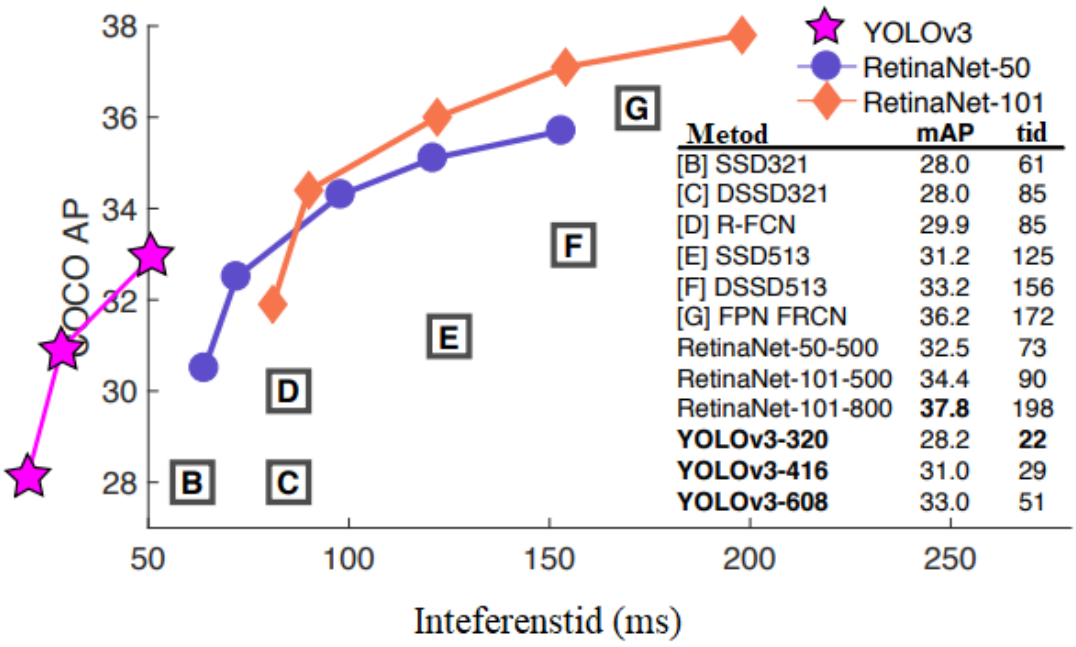
SVHN är ett dataset som består av över 600 000 bilder där varje bild innehåller ett husnummer. Dessa bilder är införskaffade från google street view och har en variation på kvalitén. Detta dataset innehåller 10 klasser där varje klass motsvarar en siffra från 0-9.

COCO datasetet innehåller färre bilder än SVHN, drygt 330 000 bilder. Däremot innehåller detta dataset 80 klasser av många olika objekt, exempelvis person, tandborste, motorcykeln mm. Vi har bara använt oss av de bilder som innehåller personer.

Anledningen till varför vi valde dessa två dataset var på grund av att dessa innehåller många bilder, vilket innebär högre precision för vår detektor när vi tränar den. Dessutom innehåller de klasser av vårt intresse: siffror och person. Detta innebar att vi behövde sammanfoga bägge dataseten till ett som ska bestå av 11 klasser: person samt 0 till och med 9.

2.3.2 YOLOv3

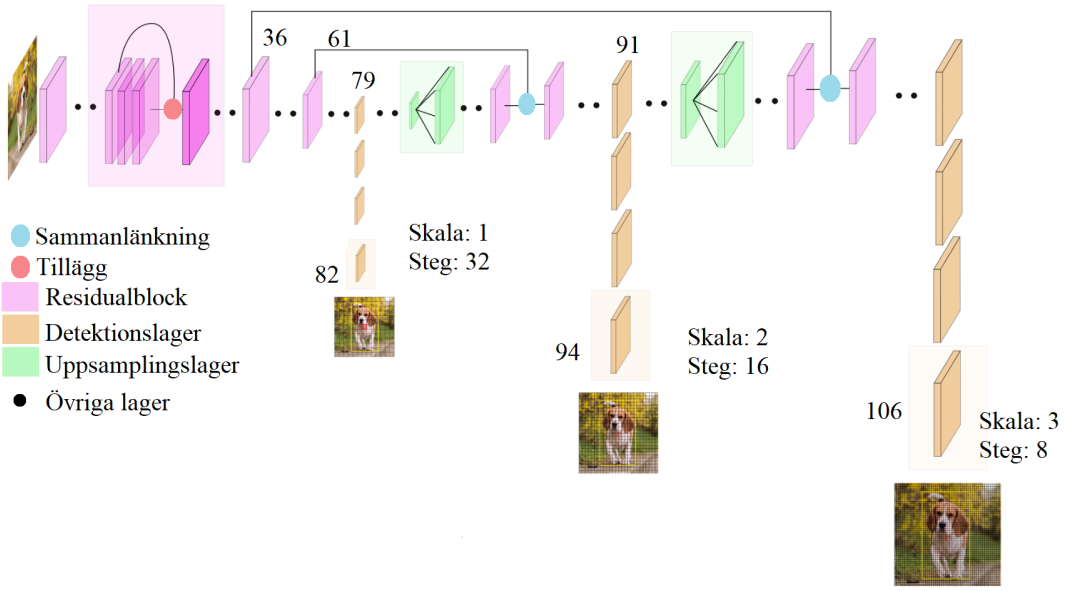
Många objektdetektionsalgoritmer använder sig utav CNN, exempelvis R-FCN, RetinaNe tit, SSD mm [9][10][11]. Bland algoritmerna sticker YOLOv3 ut i tid, detta kan ses i figur 2.2 nedan.



Figur 2.2: En jämförelse mellan olika detektor i förhållande till precision och tid. Källa: "YOLOv3: An Incremental Improvement" från databasen arXiv. Översatt till svenska.

I figur 2.2 kan man se olika objektdetektionsalgoritmers prestation där medelvärdet av precisionen plottas mot tiden som det tar för att detektera. Man kan se att YOLOv3-detektorn är den snabbaste att detektera objekt samtidigt som den har en god precision. Med detta menar skaparna av YOLOv3 att det är en "state of the art", eller en toppmodern objektdetektor [12].

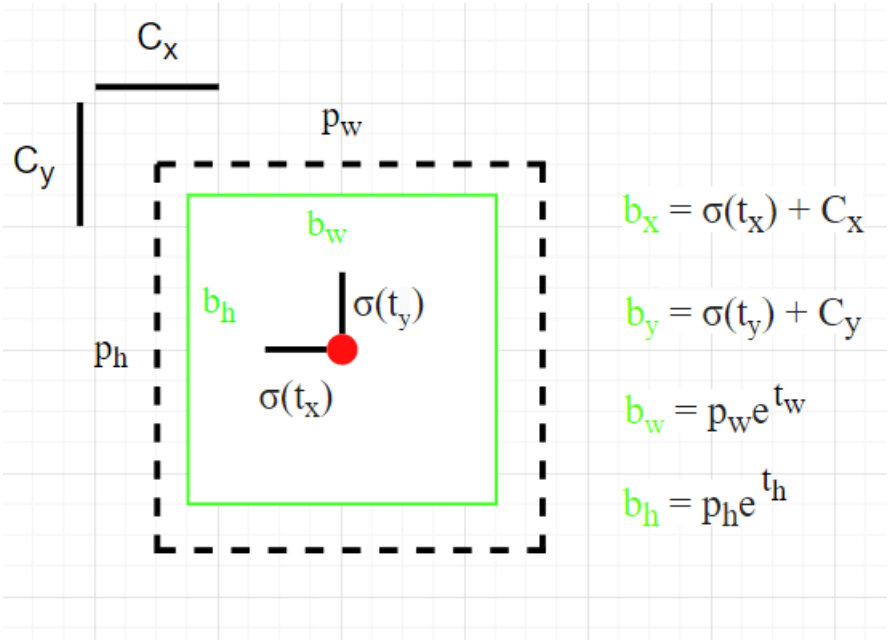
Yolov3 använder sig utav ett nätverk/arkitektur som kallas för darknet. Darknet i sig har 53 faltningslager, men för att den ska detektera läggs ytterligare 53 lager till, vilket resulterar i att totala antal faltningslager i nätverket är 106 stycken [13].



Figur 2.3: *YOLOv3 network Architecture* av Ayoosh Kathuria 2018. Återgiven med tillstånd, översatt till svenska. [14]

I figur 2.3 kan man se de lager där detektioner sker: 82:a , 94:e och 106:e (notera den gula begränsningsramen). Det som sker är att inmatningsbilden skalas ner de första 81 lagren. Sedan skapas en funktionskarta av bilden där första detektionen tar plats, därefter utsätts första funktionskartan för faltningslagren fram tills 94:e lagret där ytterligare en funktionskarta skapas som är större och har fler rutnät än den första. Samma process sker till och med den tredje. Detta leder till att det första funktionskartan har förmågan att detektera stora objekt, andra detekterar medelstora och den sista de mindre. Fördelen med denna metod är det har visats förbättra förmågan att detektera mindre objekt [15]. Denna metod kallas för FPN, engelska för *feature pyramid networks*.

Som tidigare nämnts skapas ett rutnät vid tre olika tillfällen för bilden som matas in i nätverket. Varje cell i rutnäten, oavsett storlek, förutsäger tre begränsningsramar med tre stödramar [12][16]. Det som avgör vilken cell som hanterar detektionen beror på vilken cell som centrum/mittpunkten av objektet hamnar i [17]. Den enskilde cellens uppgift är att detektera fyra koordinater: t_x, t_y, t_w och t_h. När cellen har detekterat de fyra koordinater används de tillsammans med cellens rutnätskoordinat (c_x, c_y) samt en stödrams höjd och bredd (p_h, p_w) för att skapa en begränsningsram. Detta illustreras i följande figur:

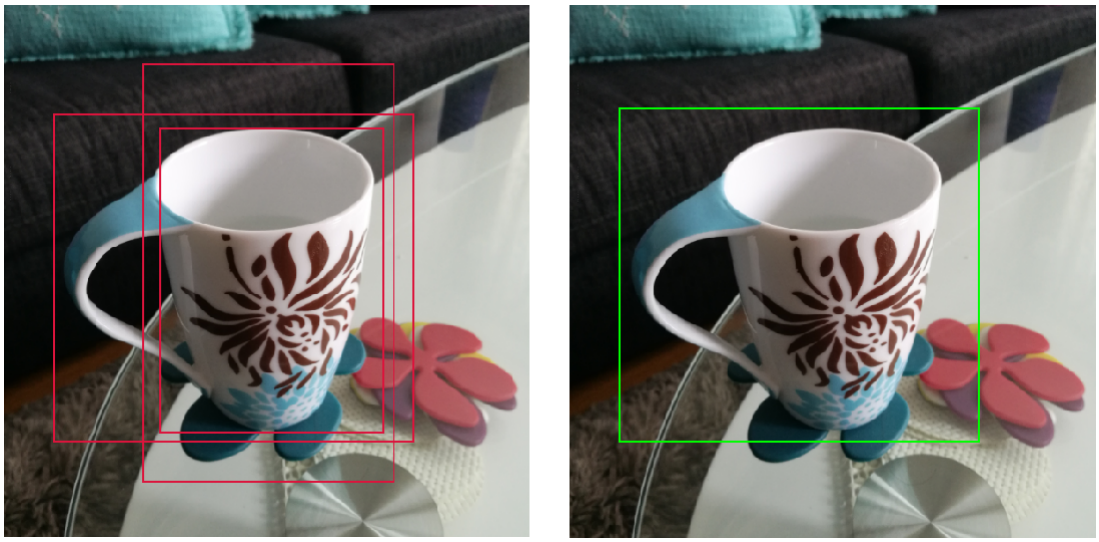


Figur 2.4: Begränsningsram med dimensioner för stödramen och förutsagda detektionen. Den streckade linjen är för stödramen, och gröna heldragna linjen för den förutsagda. Baserad på figur från [17] s. 7266.

I figur 2.4 kan man se den grundläggande principen bakom YOLOv3:s begränsningsram. Ekvationen i figuren beskriver hur nätverket transformerar dess output för att erhålla den förutsagda begränsningsramen.

När de tre begränsningsramarna har skapats av den enskilde cellen bedöms vilken av dessa som ska åskådliggöras. Detta utförs först med att titta på objektivitetsnivån som baseras på logistisk regression för varje begränsningsram, vilket innebär sannolikheten att ett objekt är inom begränsningsramen [12]. Om objektivitetsnivån inte är hög nog, över 0,5, ignoreras förutsägelsen. Det andra är att bestämma vilken klass som begränsningsramen har upptäckt, här gäller även den att med högst sannolikhet av en klass ska representeras.

Det sista steget är att behandla alla tre funktionskartor som har detekterat från inga till flera objekt. Hur går man till väga om överlappande detektioner förkommer bland funktionskartorna? YOLOv3 löser detta med hjälp av att ta kvoten mellan snittet och unionen för begränsningsramarna. Detta leder till att man får en slutgiltig begränsningsram för ett objekt. Dessa steg kallas i engelska för *Non-max suppression*, och kan illustreras enligt följande bild (figur 2.5).



Figur 2.5: Illustration av Non-max suppression för en kopp.

2.3.3 OpenCV

Open Source Computer Vision Library även kallad OpenCV ett programvarubibliotek för datorseende och maskininlärning skrivet på en öppen källkod i C och C++. OpenCV är ursprungligen utvecklat av Intel och används ofta i kommersiella syften idag. Biblioteket har mer än 2500 maskininlärningsalgoritmer samt datorsyn. Dessa algoritmer har förmågan att utföra allt ifrån att sy ihop kamerabilder till ansiktsigenkänning samt objektspårning [18].

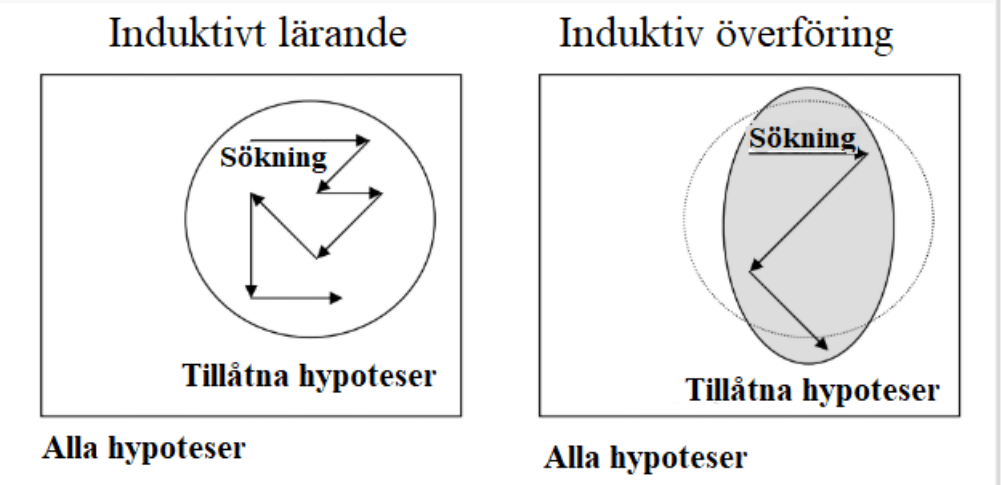
2.4 Standard maskininlärning eller djupinlärning

Standard maskininlärning och djupinlärning är olika metoder för att träna modeller som klassificerar data [19]. Den förstnämnda metoden kräver att man på egen hand för varje bild markerar ut hörn och kanter för ett objekt för att träna en modell. Modellen refererar sedan till inmatningsdatan när den ska analysera och klassificera ett objekt. I den andra metoden behöver man inte göra detta, istället matar man in bilder i ett faltningsnätverk som med sin djupinlärningsalgoritm extraherar de karaktäristiska egenskaperna hos ett objekt. Denna automatiserade egenskap hos djupinlärning medför att modeller som tränas med denna metod har en högre precision för uppgifter inom datorseende [20]. Dessutom förbättras modellen mestadels när större träningsdata är presenterad.

Med fördelarna som förekommer med djupinlärning finns det även krav för att det ska utföras. Det finns två krav som behöver fullbordas för att få en ordentlig modell som är tränad med djupinlärning. Första kravet är att ha över tusen bilder som är etiketterade, och det andra är att ha ett högpresterande grafikkort [19].

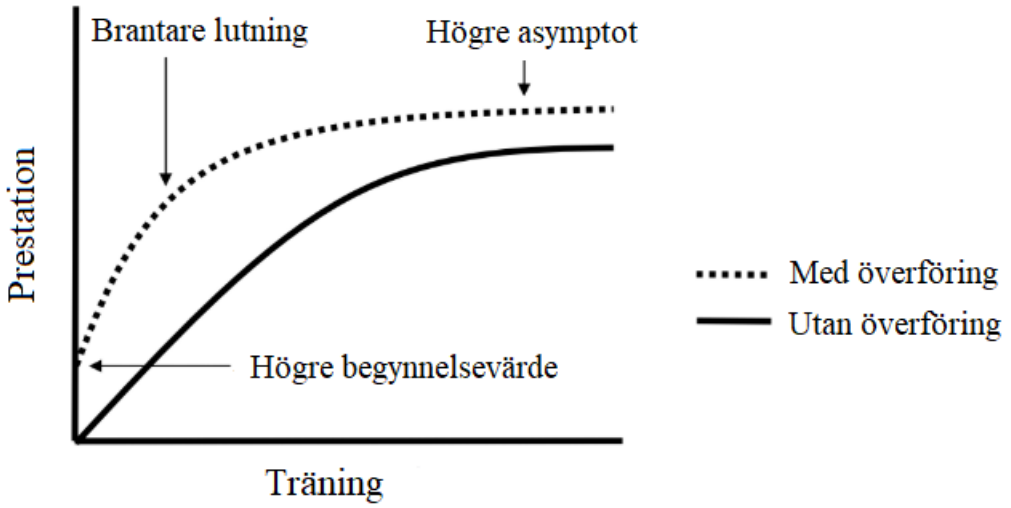
2.5 Överföringsinlärning

Överföringsinlärning (Eng. Transfer learning) är en av de tre mest vanliga metoderna som används inom djupinlärning för utföra objektklassifikation [20]. Inlärningen är en maskininlärningsteknik som baseras på att dra nytta av ett förtränat nätverk för att antingen förfina precisionen av befintliga klasser, träna nya klasser eller både och. För att använda denna metod i djupinlärning krävs det att den förtränade modellen har erhållit karaktäristiska egenskaper som är generella, dvs relaterat till målet/klassen [21]. Exempelvis är det fördelaktigt att använda sig utav ett förtränat nätverk på bilar för att lära den att identifiera lastbilar. Denna metod kallas för induktiv överföring och har fördelen att minska omfattningen av olika modeller genom att använda en modell som är menad för en annan uppgift, men som är relaterad till arbetsuppgiften [22]. Detta visualiseras i figur 2.6.



Figur 2.6: Illustration av induktiv överföring. Källa: Kapitel 11 *Transfer Learning* ur boken "Handbook of research on machine learning applications and trends: algorithms, methods, and techniques". Återgiven med tillstånd, översatt till svenska.

Eftersom överföringsinlärning bidrar till mindre omfattning av möjliga modeller minskar det även inlärningstiden som kan variera mellan minuter till timmar [20]. I samband med minskad träningstid kan även överföringsinlärning åstadkomma bättre prestation gentemot om ingen överföring hade utförts [22]. Detta illustreras av Lisa Torrey och Jude Shavlik i figur 2.7 där de redogör tre möjliga fördelar: högre begynnelsevärde av prestanda, brantare lutning på inlärningskurvan och en högre asymptot, dvs den konvergerade prestandan är bättre.



Figur 2.7: En illustrativ figur för tre fördelaktiga sätt som överföringsinlärning kan medföra. Från kapitel 11 *Transfer Learning* ur boken "Handbook of research on machine learning applications and trends: algorithms, methods, and techniques". Återgiven med tillstånd, översatt till svenska.

2.6 Homografi

Perspektiv transformation, eller homografi, är en projektiv transformation som bygger på en godtycklig homogen matris $\tilde{\boldsymbol{H}}$ med storleken 3x3 i 2D [23]. Operationen sker på homogena koordinater enligt ekvationen,

$$\tilde{\boldsymbol{x}}' = \tilde{\boldsymbol{H}}\tilde{\boldsymbol{x}},$$

där den homogena koordinaten $\tilde{\boldsymbol{x}}'$ normaliseras för att erhålla ett icke-homogent resultat $\tilde{\boldsymbol{x}}$. I 2D kan de transformerade koordinaterna uttryckas enligt ekvationerna,

$$x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}}$$
$$y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}}$$

,där h_{ij} är elementen i den homogena matrisen. Denna transformation har egenskapen att bevara räta linjer. Detta möjliggör funktionen att korrigera perspektiv för bilder, som illustreras i figur 2.8 med hjälp av OpenCV [24].

(a)

(b)

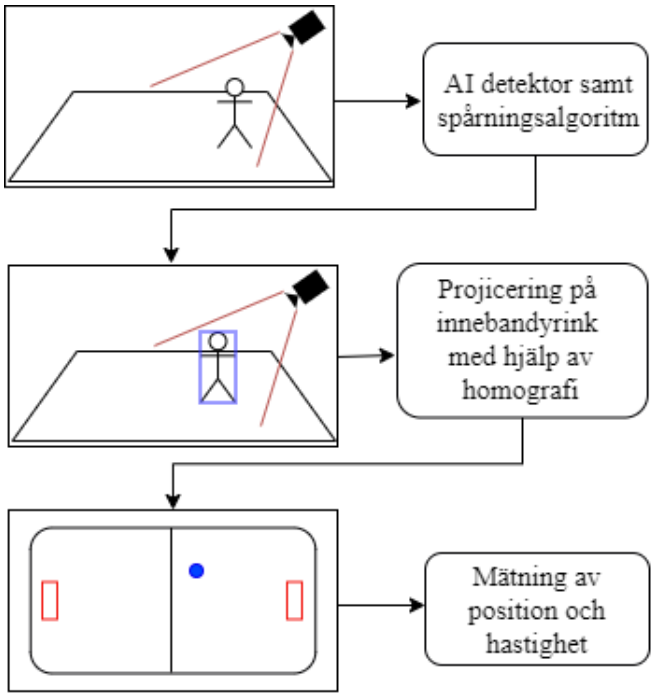
Figur 2.8: En illustration av homografi på ett område, processen går från 2.8a till 2.8b. Observera att fönstret och texten på byggnaden i figur 2.8b nu är sett ur ett rakt-framifrån-perspektiv.

3
Metod

3.1 ANNA

3.1.1 Systemet ANNA

Systemet vi har arbetat fram innehåller många delar och arbetar i flera steg, där de olika stegen har illustrerats i figur 3.1.



Figur 3.1: Illustrativ bild av hur systemet arbetar.

3.1.2 Detektor & Spårningsalgoritm

Vid objekt-detektion upptäcks ett fysiskt förmål i bilden och en ram placeras kring objekten. I vårt fall är detta en innebandyspelare, ramen som placeras runt innebandyspelare är längst ned i mitten markerad med en punkt. Det är från denna punkt vi tar ut data. Valet av punkt i mitten av nedre kant är för att när en spelare sträcker ut sina ben (vid löpning) är mätpunkten fortfarande i mitten. Detta ger bättre mätdata jämfört med exempelvis en punkt i nedre hörn. Hur detektorn sätter ut begränsningramen förklaras mer innegående i avsnitt 2.3.2.

Eftersom ANNA är ett positioneringsystem måste den även kunna spåra spelare på innebandyplanen efter detektionen. Därmed utnyttjas spårningsalogritmen Deep SORT, som står för *Simple Online and Realtime Tracking with a Deep Association Metric*, i projektet [25][26]. För att integrera Deep SORT med YOLOv3 användes tensorflow i python, med bland annat kod från github, av användarnamn Qidian213 [27]. Deep SORT-algoritmen består av flertal filer som tillsammans kan spåra flera objekt i realtid, där kalmanfilter är en väsentlig del för objektspårning.

Kalmanfiltret fungerar som en rekursiv algoritm och använder objektets tidigare position för att skapa en lämplig förutsägelse på objektets framtida position. Därav lagrar inte filtret stor mängd av data då endast den föregående positionen lagras. Algoritmen skapar en sannolikhetsfördelning för objektens position samt hastighet genom en normalfördelning. Där varje variabel (hastighet samt position) har ett eget medelvärde μ samt en varians σ^2. Eftersom hastigheten avgör hur långt objektet har rört sig råder korrelation mellan variablerna [28].

Därefter multipliceras de två erhållna täthetsfunktionerna för de två normalfördelningarna, vilket resulterar i en ny normalfördelning. Resultatet är den slutliga förutsägelsen av hastigheten och positionen för objektet. Algoritmen är en iterativ process och inläser ny inputdata samt uppdaterar de nya sannolihetsfördelningarna omgående för att spåra objektet på bilden.

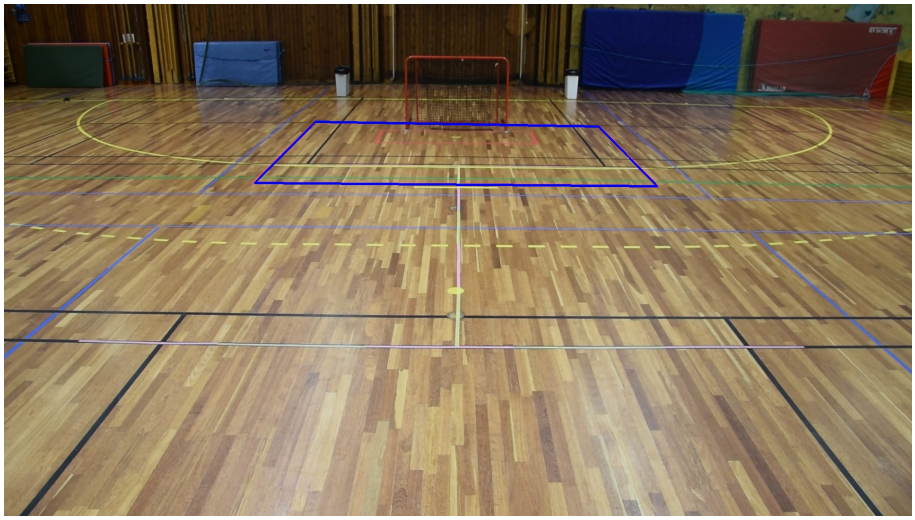
3.1.3 Träning av detektor

För att hantera filer och konfigurationer har vi använt oss av programmeringsspråket Python (3.7) med PyCharm som är en integrerad utvecklingsmiljö. När stora mängder data har behövts bearbetas så som hantering av dataseten har Google Colab använts.

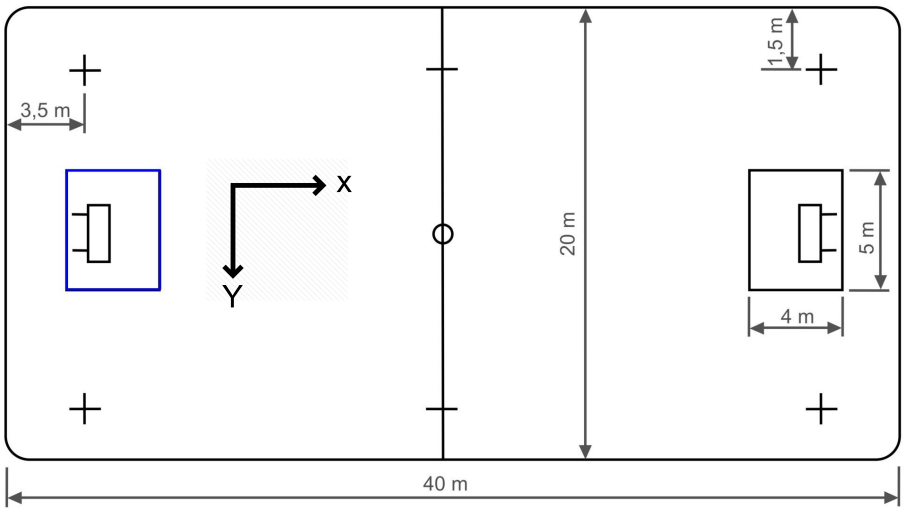
I sektion 2.5 har vi redogjort för fördelen med överföringsinlärning för djupinlärning. Denna metod har vi använt på YOLOv3 som bygger på nätverket darknet-53 och är tränad på COCO dataset. Vi konfigurerade om antalet klasser utifrån vårt intresse och tränade om nätverket så att det anpassade baserat på vår dataset. Träningen av vår modell utfördes på Google Colab som tilldelade ett högpresterande grafikkort till vår befogenhet i vår virtuella miljö.

3.1.4 Projicering på innebandyrink

För att systemet ska kunna följa en spelare projiceras den verkliga kamerabilden på ett 2D plan, se figur 3.2b. På detta sätt får vi det som om kameran hade varit fast i taket och inte filmat från sidan, se figur 3.3. Inspelningsvinkeln ändras och en spelare översätts till en punkt på skärmen. Detta görs med homografi i programvarubiblioteket OpenCV i pycharm.

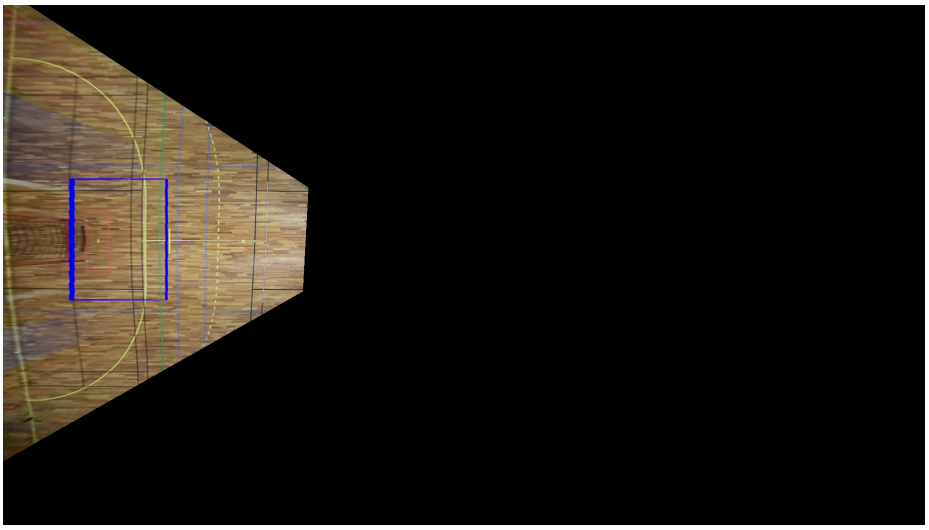


(a) Bild på innebandyplanen från kamera.



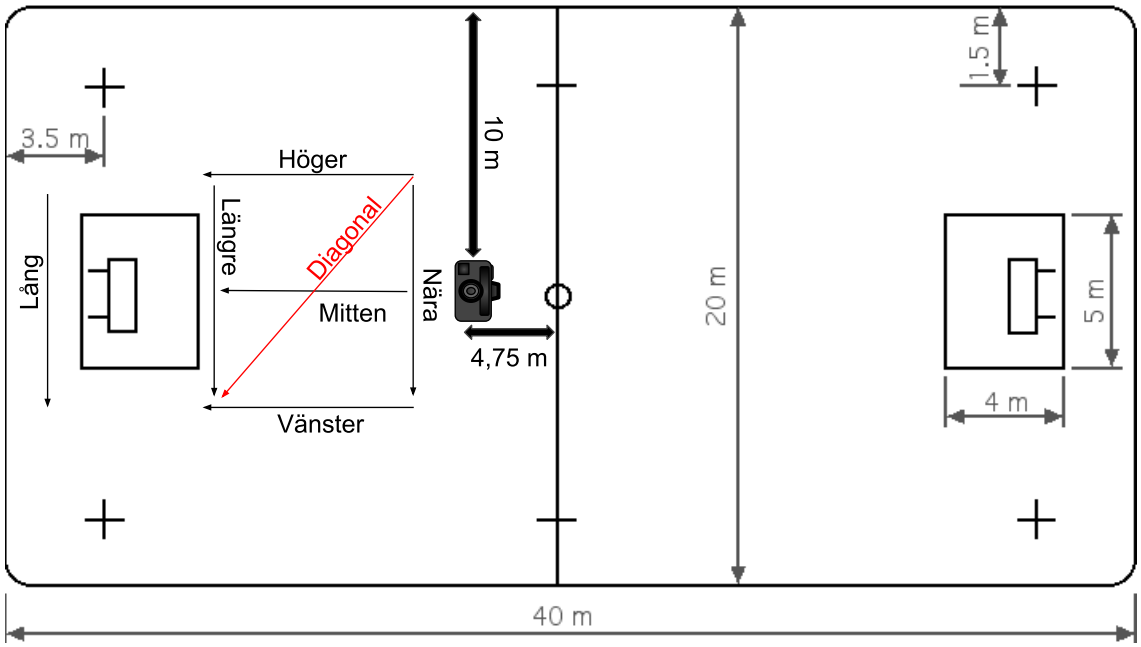
(b) Bild på planet vi vill projicera på. Bilden är tagen från wikipedia [29], dock modifierad till vår användning.

Figur 3.2: Den blåa rutan i 3.2a representerar samma blåa ruta kring målet (åt vänster) i figur 3.2b. Notera hur x- och y-axeln är orienterade. Vid hastighets- och positionsdetektion utgår vi från denna orientering.



Figur 3.3: Planet efter avbildningen, sett ifrån samma synvinkel som i figur 3.2b

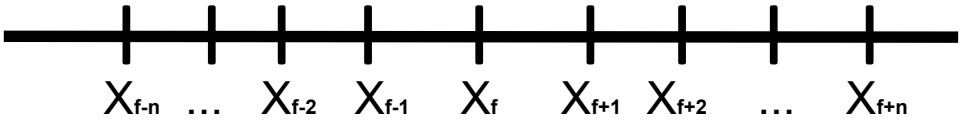
I figur 3.2a kan man se vart vi har positionerat vår kamera för att fånga våra klipp. Avståndet från den gula linjen i figuren bakom målet till vår kamera var 15,25 m, vilket är 4,75 m från att nå halva planen i x-led enligt figur 3.2b. I y-led låg kameran horisontellt längs med mittpunkten på planen, vilket var 10 m. Höjden på kameran var 2,24 m över spelplanen. I figur 3.4 illustreras uppsättningen.



Figur 3.4: En illustration för de olika löpningssträckor som användes vid framtagning av resultat. Alla löpningssträckor är 5 m långa förutom Diagonal som är 7,87 m. Sträckan Lång mättes i två olika fall, med mål (Lång_mål) samt utan mål (Lång). Bilden är tagen från Wikipedia, modifierad för att illustrera löpningarna [29].

3.2 Hastighetsmätning

Hastigheten bestämdes med hjälp av MATLAB och ett glidmedelvärde beräknades, illustrerat i figur 3.5 nedan.



Figur 3.5: En illustrativ bild över glidmedelvärde.

X_f representerar den aktuella bildrutan i videofilen. För att utnyttja ett glidande medelvärde summerar man över det önskade värdet på n, exempelvis för $n = 6$ summerar man intervallet $[X_{f-6}, X_{f+6}]$ och delar på det totala antalet element inom intervallet. Mer formellt kan vi skriva vårt glidande medelvärde för positionen $\bar{X}$ med,

$$\bar{X} = \frac{1}{2n+1} \sum_{j=0}^{2n} F(X_f - n + j,),$$

där F är en funktion som beskriver positionen för innebandy-spelaren och j representerar bildrutan.

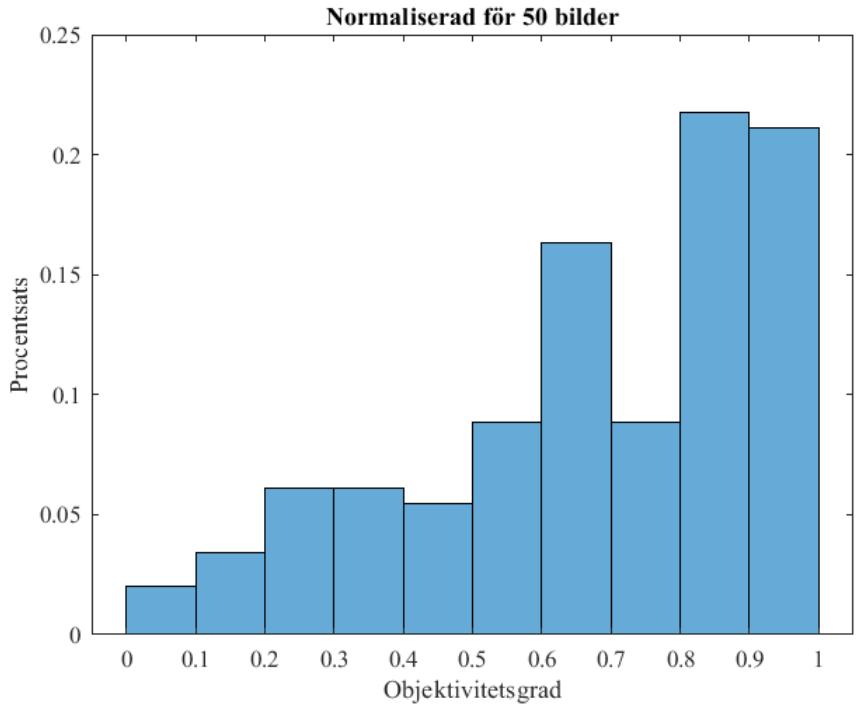
Inom projektet har intervall med $n = 6$ och $n = 12$ använts för att beräkna medelhastigheten, praktiskt innebär det att vi mäter hastigheten under ett intervall på $2n+1$ bildrutor. Då filmerna vi låtit vårt system mäta på var inspelade i 25 bilder/s motsvarar hastighetsmätningen med det glidande medelvärdet mätningar som sker varje sekund respektive halvsekund för varje bildruta. Vi valde även att använda $n = 0$.

För att bedöma vilket glidmedelvärde som var bäst anpassat för verkligheten har vi uppmätt fluktuationer i hastighet för när en person försökte springa med konstant hastighet. Detta utfördes analytiskt där personen sprang en begränsad sträcka på 5 meter. Hastigheten beräknades för varje halvmeter som jämfördes med medelhastigheten för sträckan som sprangs.

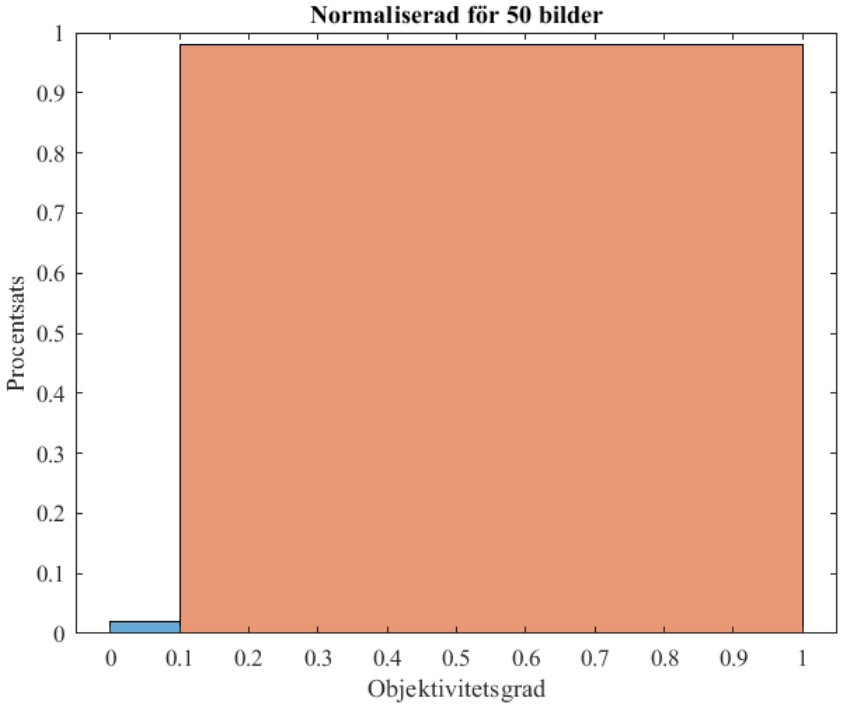
4

Resultat

4.1 Prestandadiagram



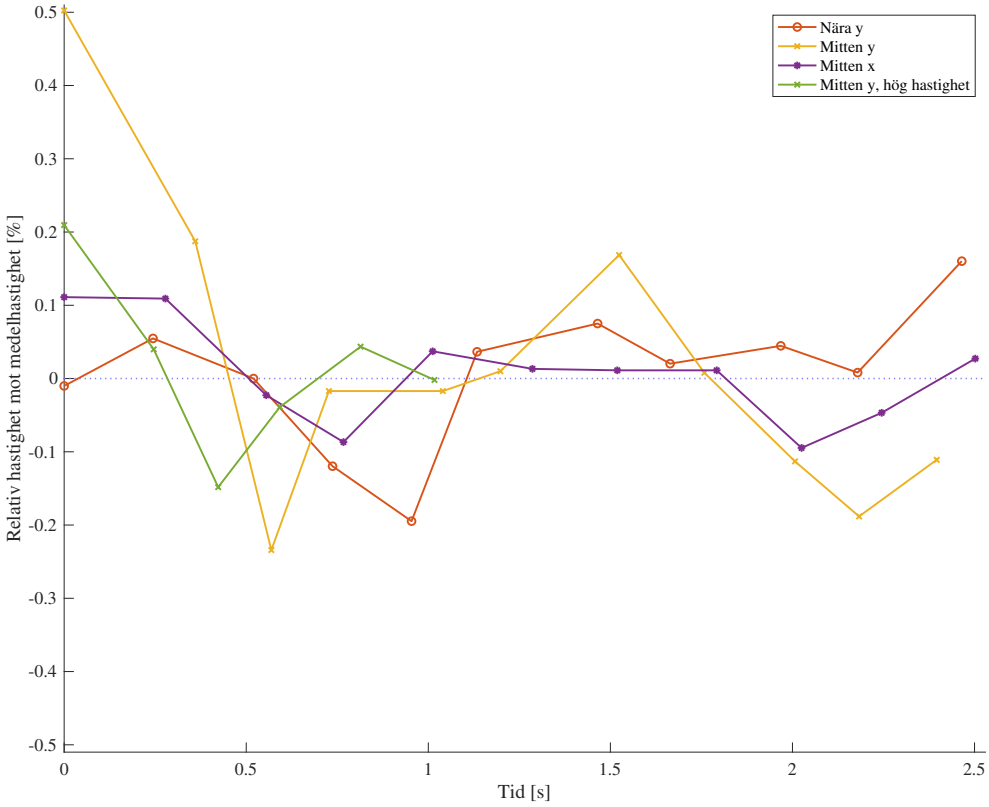
Figur 4.1: Ett histogram för 147 st normerade detektioner för klassen person i 50 bilder.



Figur 4.2: En jämförelse i signifikans med att ha en tröskel för objektivitetsgraden på 10 %.

När vår detektor var färdigtränad användes den med hjälp av en viruell maskin i Google Colab för att testa dess säkerhet på klassen person för 50 slumpmässigt valda bilder på innebandyspelare i matchscenario. Resultatet kan betraktas i figur 4.1 där alla detektioner var på människor. Eftersom ANNAs kamera kommer ha fokus på en innebandyplan försummar detta risken att detektion på felaktig person utförs. Detta gör att tröskeln för objektivitetsgraden kan minskas för att öka sannolikheten att detektera fler spelare oavsett omständigheter, exempelvis om en spelare är långt ifrån i bilden, ligger nere, osv. I figur 4.2 visualiseras den minskade tröskeln till 10 %, och man kan observera att över 99 % av detektionerna är inkluderade.

4.2 Avvikelse från medelhastigheten



Figur 4.3: I figuren visas fluktuationen av hastigheten vid analytisk jämförelse mellan medelhastigheten för löpningar av en spelare i olika omständigheter (som försöker springa med jämn hastighet). Notera att koordinaterna förhåller sig till figur 3.2b, dvs. spelaren rör sig i horisontell (y) respektive vertikal (x) riktning relativt kameran.

I figur 4.3 kan man observera fluktuationen av hastigheten relativt medelhastigheten, där hastigheten var mätt för varje halvmeter med en löpningssträcka på 5 m, notera att detta utfördes analytiskt. Bland alla omständigheter sticker den gula linjen, mitten y, ut som mest i felaktighet på max 51 % medan de resterande fallen förhåller sig maximalt runt 20 %. Att dessa fel förekommer kan bero på att löparen inte sprang med konstant hastighet. Det höga relativa felet för mitten y (löpning: Längre) kan bero av detta, dock är detta fel vid början av mätningen vilket var där spelaren tog sats.

4.3 Mätning av hastighet

Positionerna är givna från systemet och tiden av videofilerna. De genomsnittliga hastigheterna och positionerna är skalade med glidande medelvärde. All dataanalys samt plottar gjordes i MATLAB.

Vid mätning av hastigheten tog vi hänsyn till fluktuationerna som förekom, se figur 4.3. Vi valde därför att mäta hastighet över intervallet ±24 %, vilket var den näst största fluktuationen som uppmättes. Det maximala avvikelsen på 51 % förkastades eftersom vi ansåg det vara för högt. Mätningarna kan ses i tabell 4.1.

Tabell 4.1: Data från löpningar för olika orienteringar med tid angiven för att springa 5 m exkluderat diagonal som var 7,87 m. Datan visar andelen mätpunkter av hastighetsmätning som ligger inom intervallet ±24 % av medelhastigheten för respektive löpning (mätning) och glidmedelvärde n.

Löpning	Tid [s]	n = 0 [%]	n = 6 [%]	n = 12 [%]	Medelhastighet [m/s]
Diagonal	3,615	43	75	86	2,18
Höger	2,318	22	54	85	2,16
Lång	2,325	58	83	97	2,15
Lång_mål	2,419	44	89	97	2,08
Vänster	2,208	27	70	88	2,26
Mitten	2,502	34	76	82	2,00
Nära	2,465	50	77	95	2,03
Längre	2,396	71	94	94	2,09
Sprint (Mitten)	1,017	85	100	100	4,92

Tabell 4.2: En nästan identisk tabell av 4.1 dock med glidmedelvärdenas medelhastighet för de olika orienteringar.

Löpning	Tid [s]	n = 0 [m/s]	n = 6 [m/s]	n = 12 [m/s]	Medelhastighet [m/s]
Diagonal	3,615	2,68	2,22	2,21	2,18
Höger	2,318	2,20	2,32	2,35	2,16
Lång	2,325	2,10	2,12	2,14	2,15
Lång_mål	2,419	2,07	2,11	2,11	2,08
Vänster	2,208	2,18	2,18	2,18	2,26
Mitten	2,502	2,35	2,34	2,30	2,00
Nära	2,465	2,23	2,19	2,18	2,03
Längre	2,396	2,14	2,23	2,22	2,09
Sprint (Mitten)	1,017	5,17	5,01	4,82	4,92

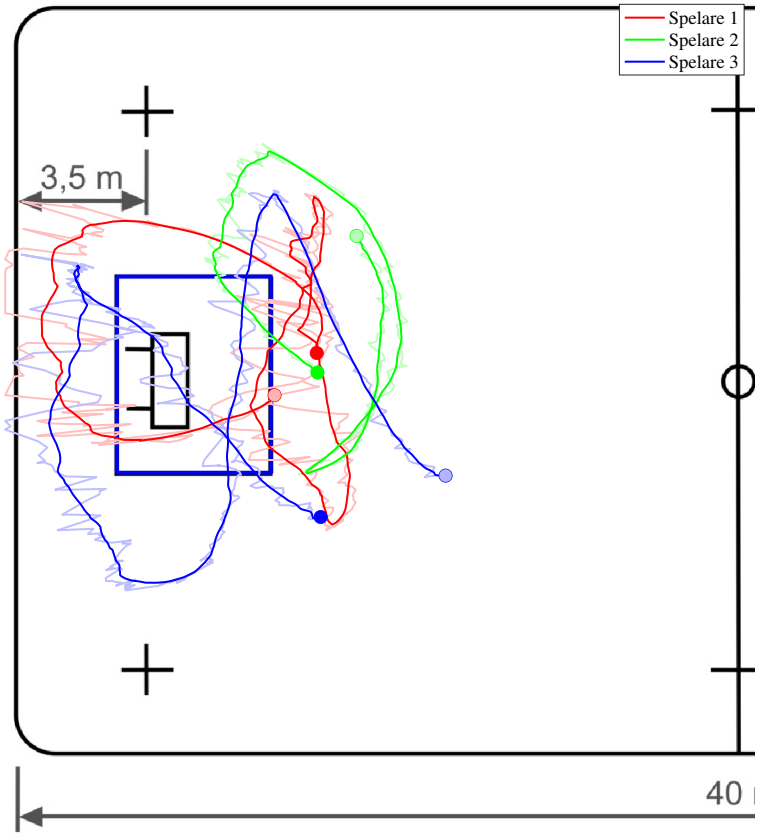
I tabell 4.2 kan man observera de olika medelhastigheterna n-värdena innehar samt den riktiga medelhastigheten för respektive löpning (mätning). I fyra av nio fall har medelhastigheten för $n = 12$ stämt bäst överens med den riktiga medelhastigheten.

För $n = 0$ stämde det bäst för tre av nio fall, $n = 6$ stämde ett fall och oavgjort för alla tre i ett fall.

Maximala relativa felet innan bearbetning med glidmedelvärde ($n = 0$) för medelhastigheten var 22,9 % och det genomsnittliga relativa felet var 7,33 %. För mätningarna med glidmedelvärde var maximala relativa felet 17 % respektive 15 % och det genomsnittliga relativa felet 5,45 % respektive 5,14 % för $n = 6$ och $n = 12$.

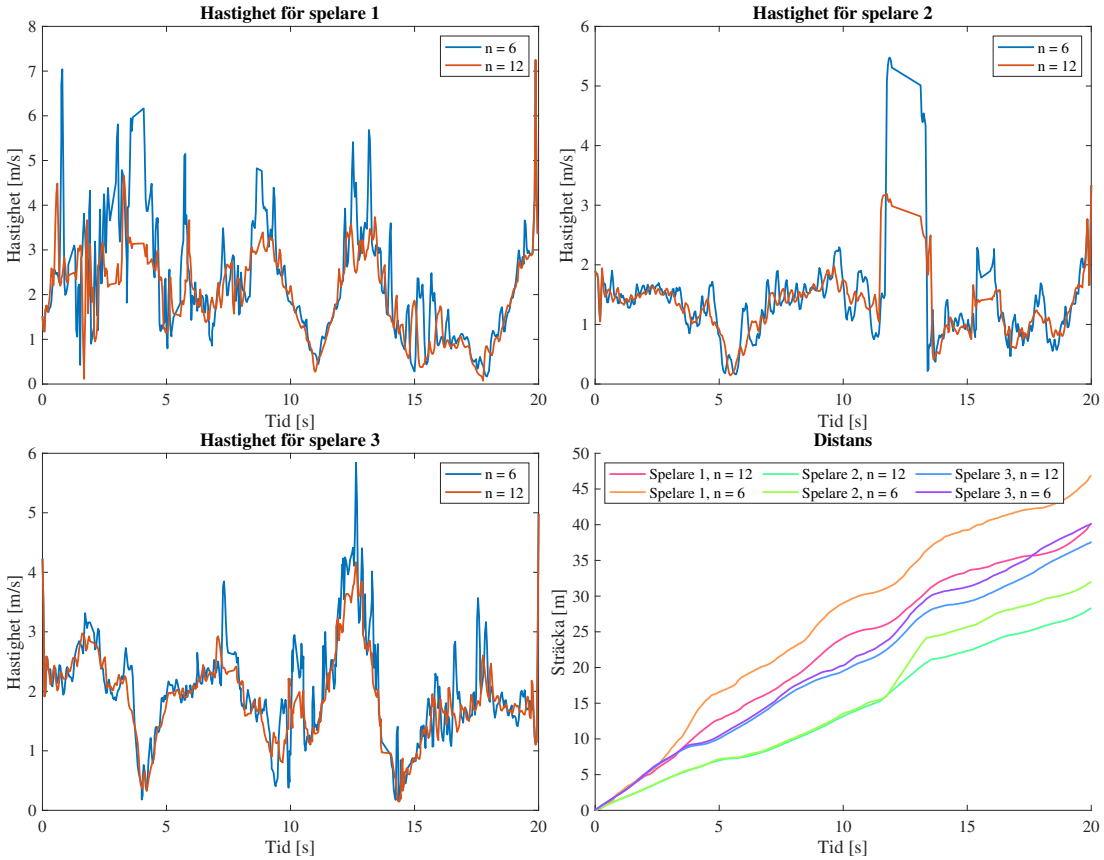
4.4 Mätning av flertal spelare

Positionerna är givna från systemet och tiden av videofilen. De genomsnittliga hastigheterna och positionerna är skalade med glidande medelvärde. All dataanalys samt plottar utfördes i MATLAB.



Figur 4.4: En avbildning av tre spelare som rör sig på innebandyplanen, där linjer med svagare färg representerar spårningen och den starkare färgen med spårningen för glidmedelvärde där $n = 12$. De ljusa punkterna är begynnelsen och mörkare slutet.

I figur 4.4 kan man se hur vårt system spårar och behandlar tre olika spelares data. De svagt färgade linjerna är koordinaterna för varje spelare som ANNA detekterar. Dessa koordinater behandlas därefter med ett glidmedelvärde, vilket syns som de starkt färgade linjerna. Man kan notera att med glidmedelvärde applicerat blir spårningen av spelare mer jämn, dvs relaterat till verkligheten, mer trovärdig.



Figur 4.5: Mätdata behandlad med glidmedelvärde i samma situation från figur 4.4 med de första tre figurerna visar varje enskild spelares hastighet mot tiden. Den sista figuren visar sträckan för varje spelare som framtogs genom integrering av respektive graf. För att sammanfoga plottarna användes MATLAB-koden subtightplot.m av Felipe G. Nievinski [30].

De tre olika spelarnas hastighet samt löpningssträcka kan observeras i figur 4.5. I de tre första hastighetsmätningarna i figuren kan de två olika intervallen för glidmedelvärden $n = 6$ och $n = 12$ jämföras. Noterbart ger lägre värde för n ett volatilt utseende, detta är en naturlig konsekvens för hur ett glidande medelvärde är konstruerat. Märkvärt är även att ett mindre intervall, $n = 6$, för det glidande medelvärdet resulterade i en längre sträcka jämfört med om ett större intervall $n = 12$ hade valts.

5

Diskussion

5.1 Diskussion kring resultatet

I figur 4.1 kan man se att vår detektor är bra på att detektera personer. Över hälften av antalet detektioner har en objektivitetsgrad större än 50 %. Eftersom vi visuellt har begränsat oss till en innebandyplan innebär detta att tröskeln för objektivitetsgraden kan minskas för att inkludera fler detektioner som representeras i figur 4.2 med en tröskel på 10 %. Även med en låg tröskel har detektorn inte misstagit ett annat objekt för en människa, vilket rättfärdigar att ha en lägre tröskel för ökat antal detektioner.

Systemet har visat sig vara bra på att mäta hastigheten för spelare, där glidmedelvärde för $n = 12$ visade sig vara bäst då minst 82 % av hastighetsmätningarna var inom ± 24 % av den verkliga medelhastigheten (vilket antas vara de verkliga värdena på fluktuation av en löpning enligt figur 4.3), se tabell 4.1. Detta visar att ett glidmedelvärde med $n = 12$ ger en bättre approximation av verkligheten gentemot $n = 0$ samt $n = 6$.

Används systemet som tänkt anser vi det vara rimligt att ha en försening på realtidsmätning med ungefär en fjärdedels sekund ($n = 6$) eller en halvsekund ($n = 12$), då data för dessa bildrutor behövs före den aktuella bildrutan som analyseras.

I figur 4.4 kan man observera trajektorian för tre spelare på en innebandyrink med glidmedelvärde $n = 0$ och $n = 12$. Resultatet blev att spelarnas trajektoria med bearbetning av glidmedelvärde på $n = 12$ var en bättre approximation av verkligheten än innan bearbetningen $n = 0$. Detta indikerar att vårt system är bra på att spåra flera spelare på planen i en film samtidigt.

I tabell 4.2 kan man jämföra fallen Längre, Lång och Nära för glidmedelvärde där $n = 12$. Tabellen visar att att Längre har ett relativt hastighetsfel på 6,22 %, motsvarande värde för Lång och Nära är 0,47 % respektive 7,3 %. Utifrån detta kan man inte med säkerhet dra slutsatsen att spelarens position på innebandyrinken har en betydelse för det relativa hastighetsfelet.

5.2 Begränsingar med ANNA

Vid konstruktionen av ANNA upptäcktes brist på stora dataset med siffror/ryggnummer. Detektorn tränades enbart på COCO datasetet med klassen person. Detektorn tränades på 64 106 bilder och kan endast känna igen människor.

Ett annat problem som väckte vår uppmärksamhet var filmningen av en hel innebandyplan med bara en kamera och ett stativ på 4 m. Vi försökte använda en kamera som kunde täcka en hel innebandyplan och valde därför GoPro Hero 5 Black. Dock kunde den enbart filma en halvplan utan direktsändning, som behövde extern hårdvara och vi fick problem med att processa videon i efterhand. Samtliga resultat bortsett från 4.1 och 4.2 har hämtats från videoklipp filmade med systemkamera.

5.3 Rekommendationer till framtida projekt

På grund av de nuvarande omständigheterna med covid-19 utbrottet har det inte varit möjligt att testa och utveckla systemet med ett innebandylag på en innebandymatch. Vi har fått nöja oss med de resultat vi har kunnat utvinna. Om detta ska testas ordentligt hade det behövts två riktiga lag med avbytare och publik.

Dessutom insåg vi inte tidigt nog projektets omfattning och vi hade behövt en bättre arbetsstruktur för att arbeta effektivare. Vi har arbetet med följande verktyg eller tjänster: Google Drive, Google Colab, Github, PyCharm (Python), Anaconda och MATLAB. Merparten av dessa var för oss helt okända och en del hade en längre inlärningskurva än förväntat. I början av projektet antog vi naivt att det största problemet skulle vara att mäta hastighet, det visade sig inte stämma då den största utmaningen för framtida projekt inom samma kategori nog kommer att vara att täcka en innebandyplan från ett kameraperspektiv. Man kan tänka sig något liknande som figur 3.3 men heltäckande. Från början hade vi strategin att kunna filma över hela innebandyplanen samtidigt med hjälp av en GoPro kamera och sedan korrigera linsförvrängningen som sker då kameran filmar med fiskögeobjektiv (bildvinkel överstigande 220 grader).

Även komplikationer med själva distortionen som förminskade planen efter korregeringsskriptet `undistort.py` kördes [32]. Eftersom den oredigerad videofilen är tangiell och radiell distortad är pixlarna inte jämt fördelade. Den icke homogena pixeldensiteten i en fiskögelins med kombination med korrigeringskoden `undistort.py` resulterar i att utsidorna av bilden har för få pixlar och fylls ut med svart. Detta ger som konsekvens att mycket av den originella bilden försvinner och endast det som är närmast kameran bibehålls.

Därefter gjordes justeringar och två mobilkameror användes istället. Videofilmerna från varje mobilkamera syddes ihop med OpenCV till en panoramafilm. På detta sätt kunde systemet prövas på en större yta än vad en enda GoPro hade kunnat

filma. Detta ledde i sin tur till fel vid detektionen då spelaren trycks ihop när man syr ihop flera bilder. Detta går säkerligen att undankomma, men vi kunde inte lösa problemet. Trots detta bör inte GoPro-idéen bortkastat helt då dess vidvinkellins är praktisk och lätt använd.

För att få en bra heltäckande bild över innebandyplanen rekommenderar vi att försöka sätta ihop en bild efter den har projicerats på en karta (ex figur 3.2b) och att även utföra spårningen då.

Nuvarande system kan köras live med ca 7 bilder/s, detta anser vi vara för lågt då vi vill ha upp till ca 25 bilder/s för att få liknande resultat som vi producerat då vi lät systemet granska en video. Vi har använt oss av grafikkortet Nvidia GTX 1070 då det använder CUDA som låter det bearbeta data parallellt och på så vis snabbare. För att öka antalet bilder/s kan ett bättre grafikkort användas. Detta är en enkel men ekonomiskt kostsam metod för att öka antalet bilder/s. En annan lösning till detta är att använda sig av en snabbare detektor, vi använde oss av YOLOv3. Det finns en snabbare detektor med sämre precision som heter YOLOv3-tiny. Vi använde denna detektor men utan vidare framgång då vi fick en ökning till enbart 9 bilder/s, vi kan dock inte svara på varför vi inte fick en större ökning, en större ökning var förväntad. I dagsläget har en ny detektor släppts, YOLOv4, vi råder er även att använda denna istället för v3 (v4 släpptes i april och vi hade inte tid att byta). Vi har använt oss av två olika dataset för att träna vår detektor, COCO och SVHN. COCO för människoigenkänning och SVHN för nummer på ryggen, den senare till vårt misslyckande då kunskapen om inlärning hos AIn inte var tillräcklig. Idealt skulle det vara bra att ha ett dataset med bara innebandyspelare (i matchställ). Då skulle man även få med deras siffror.

5.4 Felkällor

Resultatet för hastighetsmätningen har jämförs med ett uppmätt värde för den riktiga hastigheten, vilket innebär ett mänskligt fel. Den riktiga hastigheten som används för referens är beräknad på så sätt att spelaren springer en känd sträcka och tiden mäts med ett tidtagarur. Detta kan ha bidragit till fel av beräkningen för det relativa felet för medelhastigheterna i sektion 4.3. Däremot bör inte detta ge missvisande resultat då det råder om en felmarginal på en tiondels sekund.

Eftersom hastigheten är tidsderivatan av sträckan beror den på hur bra systemet är på att positionera spelaren. Fel i hastighetsmätning kommer då positionen inte kan detekteras exakt. Detta kan bero på flertal olika anledningar som exempelvis att bilden som projiceras har en linsförvrängning, denna behöver inte vara stor men kan ändå påverka resultatet då homografin antas ha en bild utan linsförvridning.

Ett problem som förekommer vid spårningen är bildningen av begränsningsramen. Begränsningsramens storlek kan variera och detta sker i stor utsträckning när spelare korsar varandra. Exempelvis kan en spelare täcka en annan så att bara en del

av överkroppen syns, systemet kan fortfarande detektera att det är en spelare men ser inte benen och sätter en begränsningsram på bara överkroppen. Detta gör att begränsningsramen ger missvisande värden på var spelaren befinner sig då mätpunkten för projektion är i mitten av nedre kant på ramen. En möjlig lösning till det specifika problemet med spelare som korsar varandra är att sätta kameran högre upp för att få en snävare vinkel, vilket minskar problemet med att begränsningsramen klipps av då spelare korsar varandra. Då vi positionerat kameran med en höjd på 2,24 m förekommer detta fel, men uppskattningen av felet är svårbedömd. För att hantera det generella problemet med att beräsningsramens storlek varierar har glidmedelvärde använts, till vår bedömning, mycket framgångsrikt.

Springer en spelare fort finns det risk att spelaren på bilden som tages till videon blir suddig och att detektorn inte kan se någon spelare. Hastigheten spelaren då måste ha måste överskrida den maximala hastighet som redovisas i tabell 4.1 och 4.2.

6
Slutsats

ANNA är en prototyp som visar möjligheterna för att bygga enkla system som är välgrundade när det kommer till spårning av spelare och dess hastighet. Trots detta är systemet inte idealt då det klart existerar flera utvecklingsområden. Med en bättre strategi samt planering för systemet kan flera av de problem vi upptäckte under arbetsgången åtgärdas. Däremot har systemet ANNA visat sig vara ett bra positioneringsystem då hastigheten samt positionen för varje spelare på planen kan avgöras med bra precision.

Positioneringssystemet tränades på två olika dataset COCO samt SVHN, där den sistnämnde inte gav något användbart resultat. På grund av brist på dataset kunde inte spelare identifieras med lagtröja eller lagnummer. Trots detta kan spelare urskiljas med ett unikt id. För framtida projekt rekommenderas träning på dataset som kan användas för att skilja på lagen på ett smidigt sätt. Därav bör inte SVHN bortkastas helt, då detta är ett bra dataset som innehåller nummer och kan användas för att urskilja spelare med lagnumer. En annan åtgärd är att skapa ett eget dataset och därmed anpassa träningen helt efter sin egna preferens.

I sektion 4.1 kan man observera objektivitetsgraden samt procentsatsen för detektion. Resultatet visar att en lägre objektivitetsgrad på detektorn ger fler detektioner. Genom att sätta en lägre objektivitetsgrad för klassificering förloras inte spelare på planen. Därav är det rimligt att ha ett lågt tröskelväde på detektorn, då det endast kommer att finnas spelare på planet. På grund av detta kan systemet utvinna det mest ideala resultaten, där varje spelare på innebandyplanen är identifierad och spårbar.

Hastighetsmätningen för spelare på innebandyplanen gjordes med hjälp av glidande medelvärde. Resultatet visar att det är en bra metod att använda för att erhålla medelhastigheten hos en spelare. I våra nio mätningar i tabell 4.1 presterade mätningar med glidmedelvärde för $n = 12$ bäst med maximala relativa medelhastighetsfelet 15 % och genomsnittliga relativa medelhastighetsfelet 5,14 %.

Det största problem som upptäcktes under arbetetsgången var projektionen för en hel innebandyplan, då vår kamerautrustning inte var tillräcklig för att täcka en 20x40 m^2 plan. En möjlig åtgärd för detta problem är att ta till fler kameror och därefter sy ihop bilderna i OpenCV.

Trots alla komplikationer och motgångar vi har upplevt under projektets gång har

vi i slutändad kunnat bygga ett positioneringsystem, vilket identifiera en människa med 99 % sannolikhet vid objektivitetsgrad på 10 %. På denna grund är detta arbete ett bra underlag för framtida projekt.

Litteraturförteckning

[1] Intel True View https://www.intel.com/content/www/us/en/sports/technology/true-view.html

[2] Magnus Karlsteen, Jonathan Weidow TXF04-20-01 :Innebandy-analys av spelares rörelsemönster,http://www.chalmers.se/SiteCollectionDocuments/Fysik/Student%20projects/Kandidatarbeten%202020/TIFX04-20-01.pdf , Hämtad 20 Januari 2020.

[3] Computer Sweden, "Faltningsnätverk", *IT-ORD* [Hemsida], https://it-ord.idg.se/ord/faltningsnatverk/ , Hämtad 10 April 2020.

[4] Harshita Srivastava,"Convolutional Neural Networks Explained", *Magoosh Data Science Blog*, APRIL 24 2018.https://magoosh.com/data-science/convolutional-neural-networks-explained/, Hämtad 15 April 2020. Web.

[5] K. Fukushima, S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition." *Competition and cooperation in neural nets*. Springer, Berlin, Heidelberg, 1982. 267-285.

[6] Harsh Pokharna,"Pooling Layers"*Medium*[Hemsida],https://medium.com/technologymadeeasythe-best-explanation-of-convolutional-\neural-networks-on-the-internet-fbb8b1ad5df8

[7] "Om dataset". *Sveriges riksdag*, (n.d) https://data.riksdagen.se/dokumentation/om-dataset/, hämtad 17:e april 2020.

[8] Netzer, Yuval, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng "Reading Digits in Natural Images with Unsupervised Feature Learning", *NIPS Workshop on Deep Learning and Unsupervised Feature Learning* 2011.

[9] Dai, Jifeng, Yi Li, Kaiming He, Jian Sun "R-FCN: Object Detection via Region-based Fully Convolutional Networks", *Advances in neural information processing systems*. 2016.

[10] Lin, Tsung-Yi, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollár "Focal Loss for Dense Object Detection." *Proceedings of the IEEE international conference on computer vision*. 2017.

[11] Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg "SSD: Single Shot MultiBox Detector." *European conference on computer vision*. Springer, Cham 2016.

[12] Joseph Redmon, Ali Farhadi “YOLOv3: An Incremental Improvement”, *arXiv*, 2018.

[13] Redmon, Joseph “Darknet: Open Source Neural Networks in C.” `http://pjreddie.com/darknet/`, 2013–2016.

[14] Kathuria, Ayoosh. “What’s new in YOLO v3?”, *Towards data science*, Medium, 2018, `https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b`.

[15] Lin, Tsung-Yi, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, Serge Belongie “Feature Pyramid Networks for Object Detection”, *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

[16] “Anchor Boxes for Object Detection”, (n.d), hämtad från `https://se.mathworks.com/help/vision/ug/anchor-boxes-for-object-detection.html`, hämtad maj 2020

[17] Redmon, Joseph, Ali Farhadi “Yolo9000: Better, faster, stronger”, *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[18] OpenCV , ”About”, *OpenCV* `https://opencv.org/about/`, Hämtad 14 April 2020. Web.

[19] “Introduction to Deep learning: Machine Learning vs Deep Learning”, 2017, hämtad från `https://se.mathworks.com/videos/introduction-to-deep-learning-machine-learning-vs-deep\-learning-1489503513018.html`.

[20] “What is Deep Learning?”, (n.d), hämtad från `https://se.mathworks.com/discovery/deep-learning.html` maj 2020.

[21] Yosinski, Jason, Jeff Clune, Yoshua Bengio, Hod Lipson “How transferable are features in deep neural networks?.” *Advances in neural information processing systems*. 2014.

[22] Torrey, Lisa and Jude Shavlik. “Transfer Learning.” *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI Global, 2010. 242-264.

[23] Szelski, Richard. *Computer Vision: Algorithms and Applications*. Springer Science Business Media, 2010.

[24] OpenCV ”Basic concepts of the homography explained with code” *OpenCV*, (n.d) `https://docs.opencv.org/master/d9/dab/tutorial_homography.html` , Hämtad 4 Juni 2020

[25] Wojke, Nicolai, Bewley, Alex and Paulus, Dietrich. ”Simple Online and Realtime Tracking with a Deep Association Metric” *2017 IEEE International Conference on Image Processing (ICIP)*, IEEE, DOI : 10.1109/ICIP.2017.8296962 , 2017, 3645–3649

[26] Wojke, Nicolai and Bewley, Alex. ”Deep Cosine Metric Learning for Person Re-identification”*2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE , DOI: doi=10.1109/WACV.2018.00087, 2018, 748–756

[27] Qidian213, deep_sort_yolov3, (ND) hämtad från `https://github.com/Qidian213/deep_sort_yolov3`. April 2020

[28] Babb, Tim, ”How a Kalman filter works, in pictures”, *bzarg*, Augusti 2015. `https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/`. Hämtad 13 Maj 2020. Web.

[29] Wikipedia, “Floorball” `https://en.wikipedia.org/wiki/Floorball#/media/File:Floorball_rink1.svg` , Hämtat Maj 2020.

[30] Nievinski, G, Felipe, ’Subtightplot’ (ND), hämtad från `https://www.mathworks.com/matlabcentral/fileexchange/39664-subtightplot`, Hämtat Maj 2020.

[31] Mordvintsev, Alexander and Abid K. ”Camera Calibration”, *OpenCV.* , `https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_calibration/py_calibration.html`. 2013. Web

[32] Williams David, Undivid,`https://github.com/cdw/undivid`2014

[33] Wikipedia, ”Checkerboard”, `https://en.m.wikipedia.org/wiki/Checkerboard#/media/File%3AFont_Awesome_5_solid_chess-board.svg`.2018. Web

A
Appendix

A.1 Arbetsfördelning

Namn	Arbetstid	Arbetsuppgifter (fördelning inom gruppen)
Amina Warsame	323 h	korrigering av linsdistortion (100 %) rapportskrivande (40 %) praktiskt arbete (30 %)
Kevin Vu	327 h	positionsbestämning (kod, 30 %) resultatberäkningar (10 %) praktiskt arbete (20 %) hantering av dataset (10 %) rapportskrivande (40 %)
Nicklas Lindevall	356 h	AI träning (100 %) positionsbestämning (kod, 70 %) resultatberäkningar (90 %) rapportskrivande (20 %) praktiskt arbete (50 %) hantering av dataset (90 %) projicering (100 %)

A.2 Kamerakalibrering

Kameran som användes vid projektet är en GoPro Hero 5 Black. Denna kameran har både radiell och tangiell distortion, vilket resulterar i att raka linjer blir böjda och att vissa delar i bilden verkar närmare än andra. För att undankomma dessa förvrängningar behöver man fem parametrar som kallas för distortionskoefficienterna. Dessa hittas genom kalibrering med ett väldefinierat mönster och OpenCV [31]. I vårt fall användes ett schackmönster med 8x8 rutor som referens, se figur A.2.

Korrektionen för den tangentiell distorsionen kan beskrivas med ekvationen,

$$X_{corrected} = x + 2p_1xy + p_2(r^2 + 2x^2)),$$
$$Y_{corrected} = y + p_1(r^2 + 2y^2) + 2p_2xy),$$

Korrektionen för den radiella distorsionen kan beskrivas som,

$$X_{corrected} = x(1 + k_1r^2 + k_2r^4 + k_3r^6)$$

$$Y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

,där $= (k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3)$ är distortionskoefficienter.

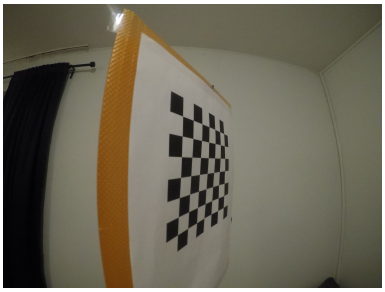
För att kalibrera kameran behöver man även de intrinsiska och extrinsiska kameraparametrarna som är specifika för kameran. De intrinsiska kameraparametrarna kallas även för kameramatris och beskrivs av en 3x3 matris,

$$\text{Kameramatris} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

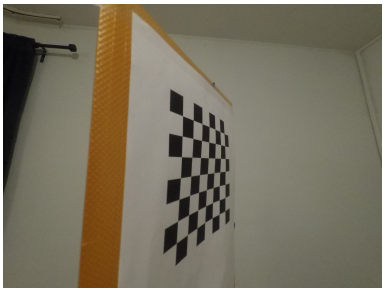
,där (f_x, f_y) beskriver kamerans fokallängd och (c_x, c_y) beskriver kamerans optiska centra.

Parametrarna beskriver rotation och translationsvektorerna som används för översättningen mellan kamerans referensplan och den yttre verkligheten. Dessa hittas med hjälp av schackmönstret i samband med OpenCV funktionen `cv2.findChessboardCorners` för Python [31].

Efter kamerakalibreringen används `Undistort.py` för att korregera förvrängningen [32], se figur A.1.



(a) Innan korrigering



(b) Efter korrigering med `Undistort.py`

Figur A.1: Förvrängningen i den radiella och den tangentiell riktningen syns klart i bild (a), där närmare föremål böjs och storleken på objekten inte representeras korrekt. Efter korrigeringen i bild (b) kan man observera att bilden har rätats ut vilket resulterar i en bild som liknar verkligheten.

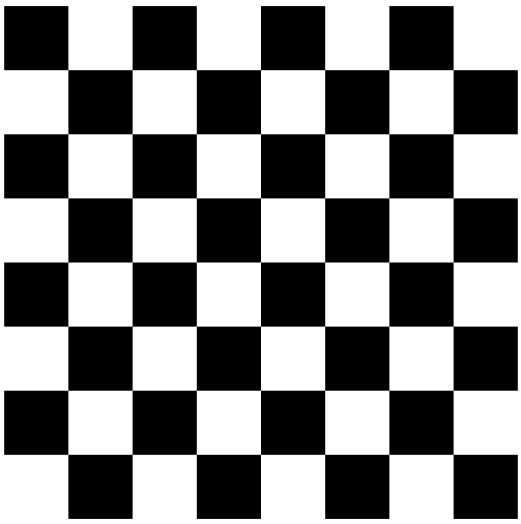
A.2.1 Kamermatris och Distortionkoefficienter

Efter kalibrering med 50 bilder på en 8 x 8 schackbräda kunde dessa kameraparametrar och matriser erhållas,

$$\text{Distortionskoefficienter} = \begin{bmatrix} -0,2761167554090371 \\ 0,12063701031093957 \\ 0,00090502852319349 \\ -0,00011988037639007054 \\ -0,029483225631880928 \end{bmatrix} = \begin{bmatrix} k_1 \\ k_2 \\ p_1 \\ p_2 \\ k_k \end{bmatrix}$$

$$\text{Kameramatris} = \begin{bmatrix} 1749.5895302743875 & 0 & 1988.258197104154 \\ 0 & 1741.4984080736551 & 1490.7255221019393 \\ 0 & 0 & 1 \end{bmatrix}$$

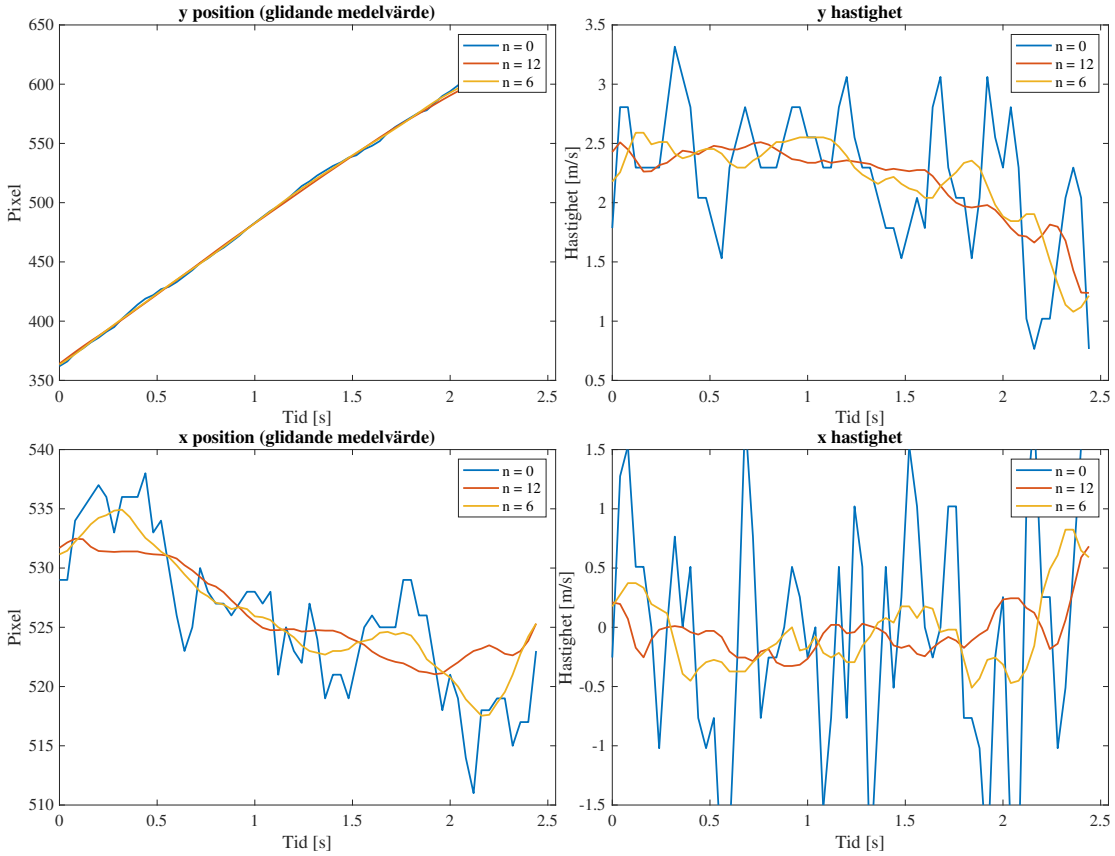
vilket direkt kunde användas i `Undistort.py` för att korrigera förvrängningen. Schackmönstret som användes för kalibreringen kan observeras i figur A.2.



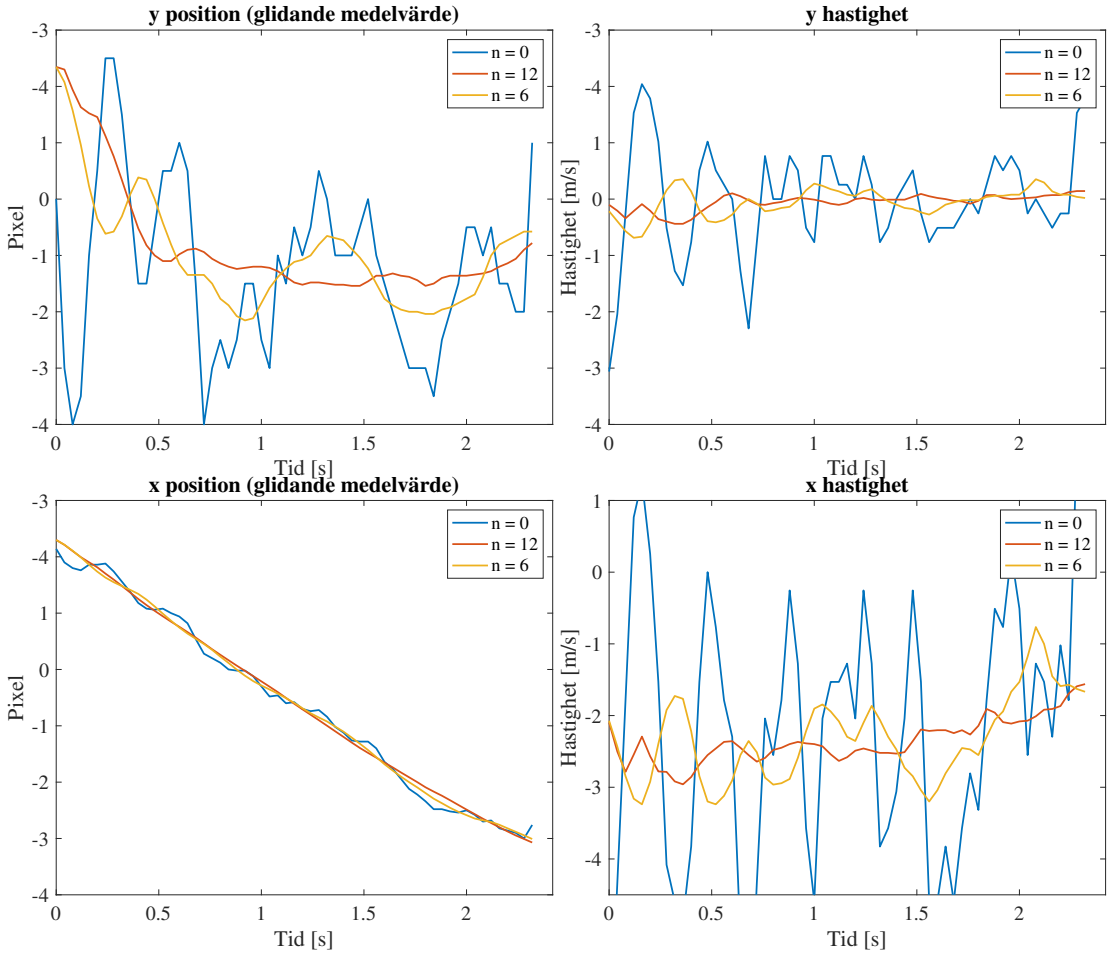
Figur A.2: Ett 8 x 8 schackpjäs mönster som användes vid referens för kamerakalibreringen, hämtad från wikipedia [33]

A.3 Hastighetsanalys

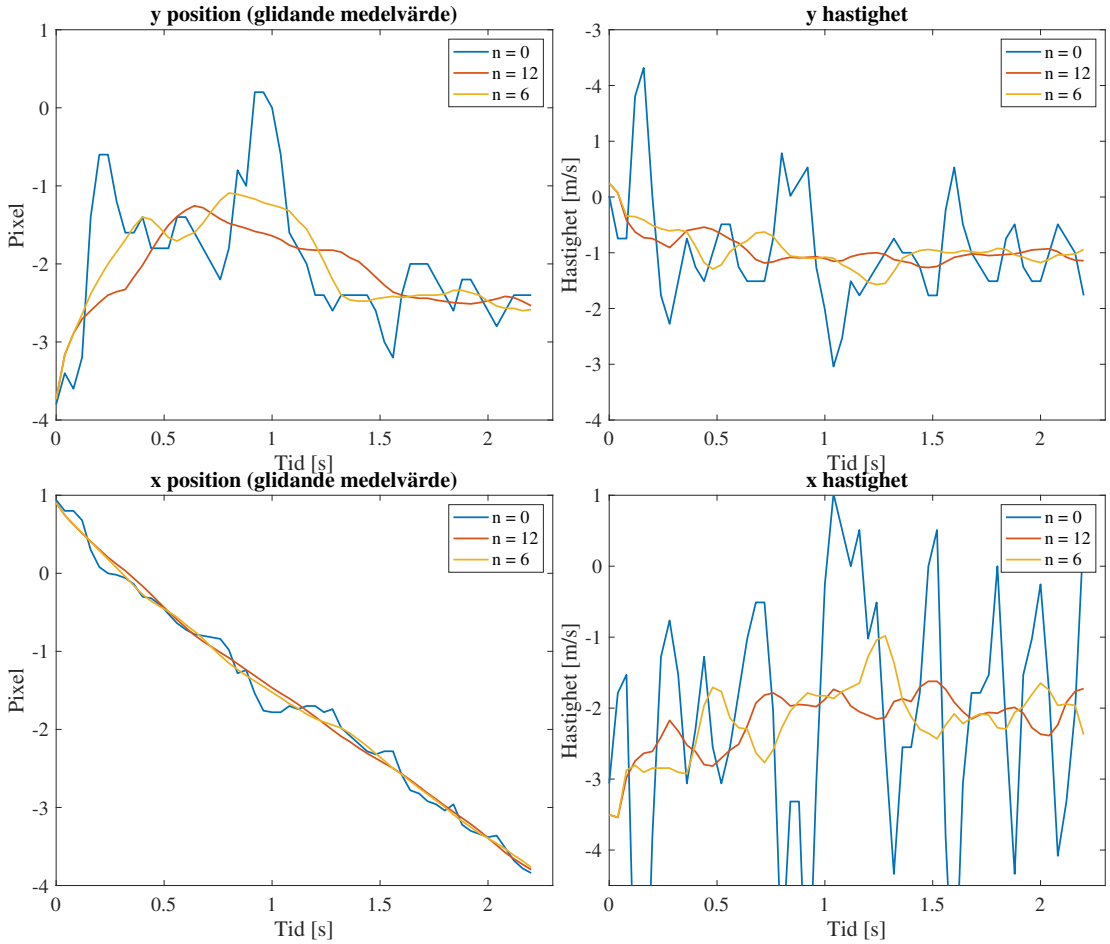
Analysen av all data gjordes i MATLAB, med tiden given från videofilen samt positionen given från ANNA. Löpningssträckorna i följande grafer är illustrerade i figur 3.4.



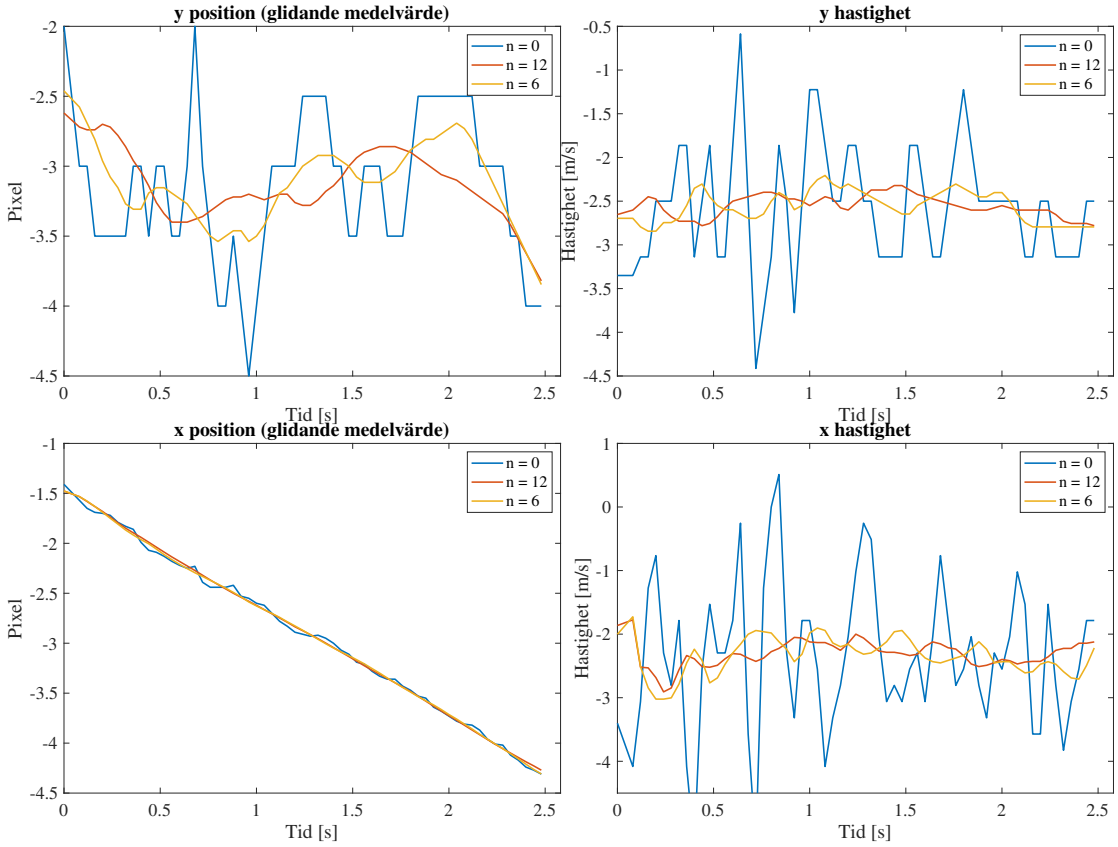
Figur A.3: En spelares rörelse på en 5 m sträcka, definierad som Nära, tagen från en videofil. De olika färgade linjerna representerar de olika intervallen som valdes för det glidande medelvärdet.



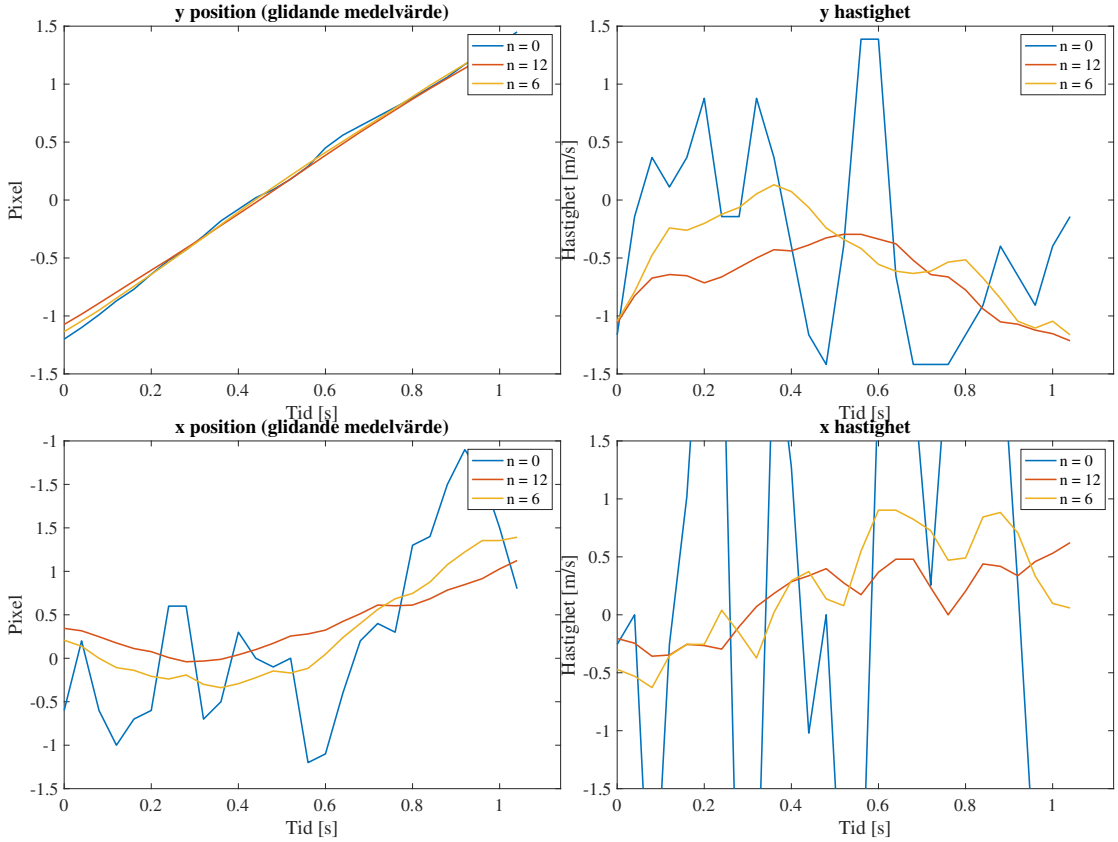
Figur A.4: En spelares rörelse på en 5 m sträcka, definierad som Höger, tagen från en videofil. De olika färgade linjerna representerar de olika intervallen som valdes för det glidande medelvärdet.



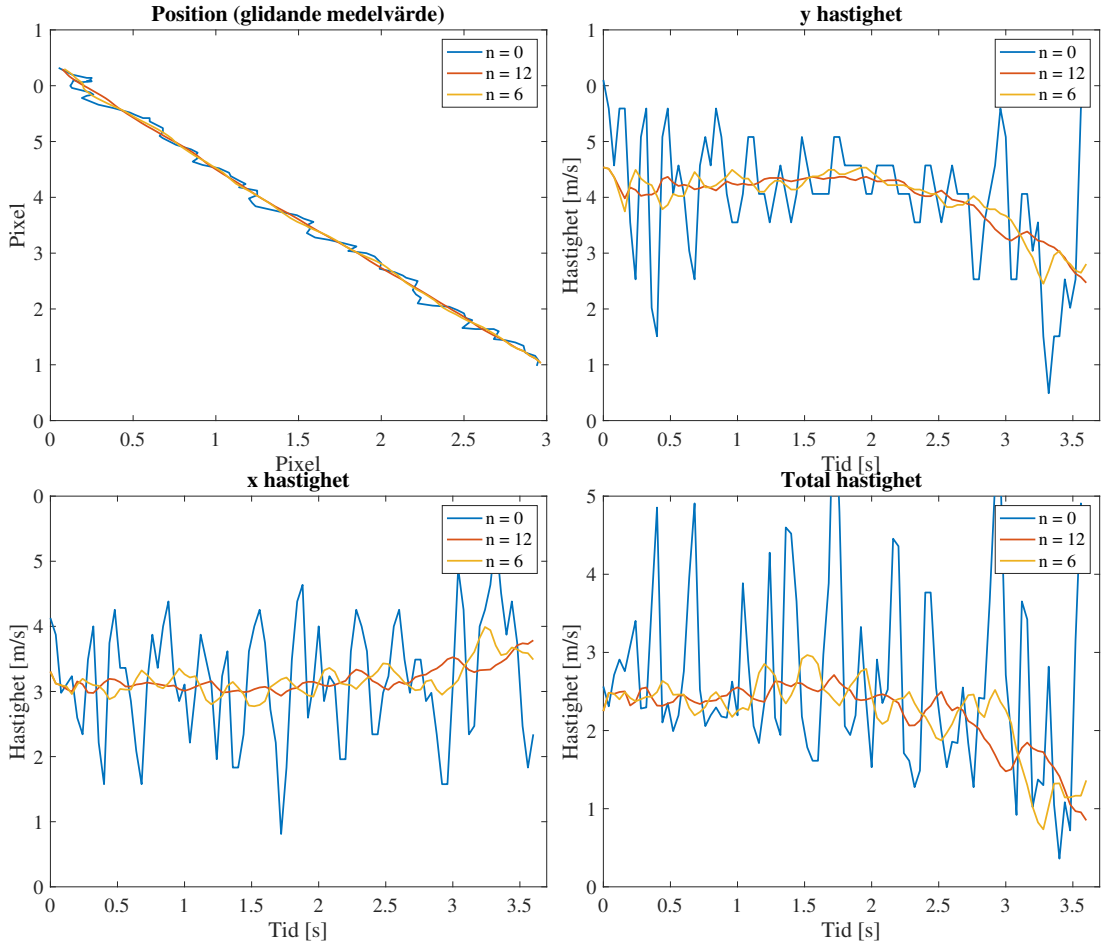
Figur A.5: En spelares rörelse på en 5 m sträcka, definierad som Vänster, tagen från en videofil. De olika färgade linjerna representerar de olika intervallen som valdes för det glidande medelvärdet



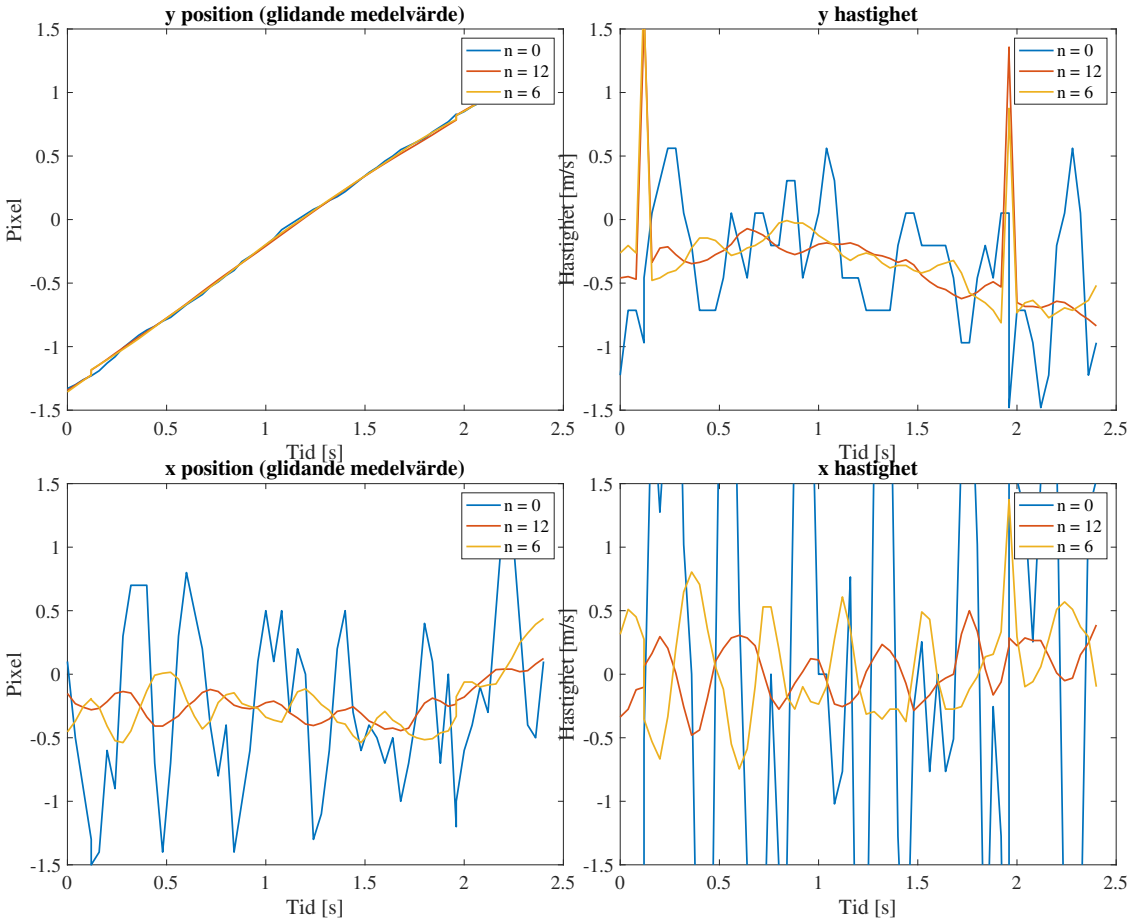
Figur A.6: En spelares rörelse på en 5 m sträcka, definierad som Mitten, tagen från en videofil. De olika färgade linjerna representerar de olika intervallen som valdes för det glidande medelvärdet.



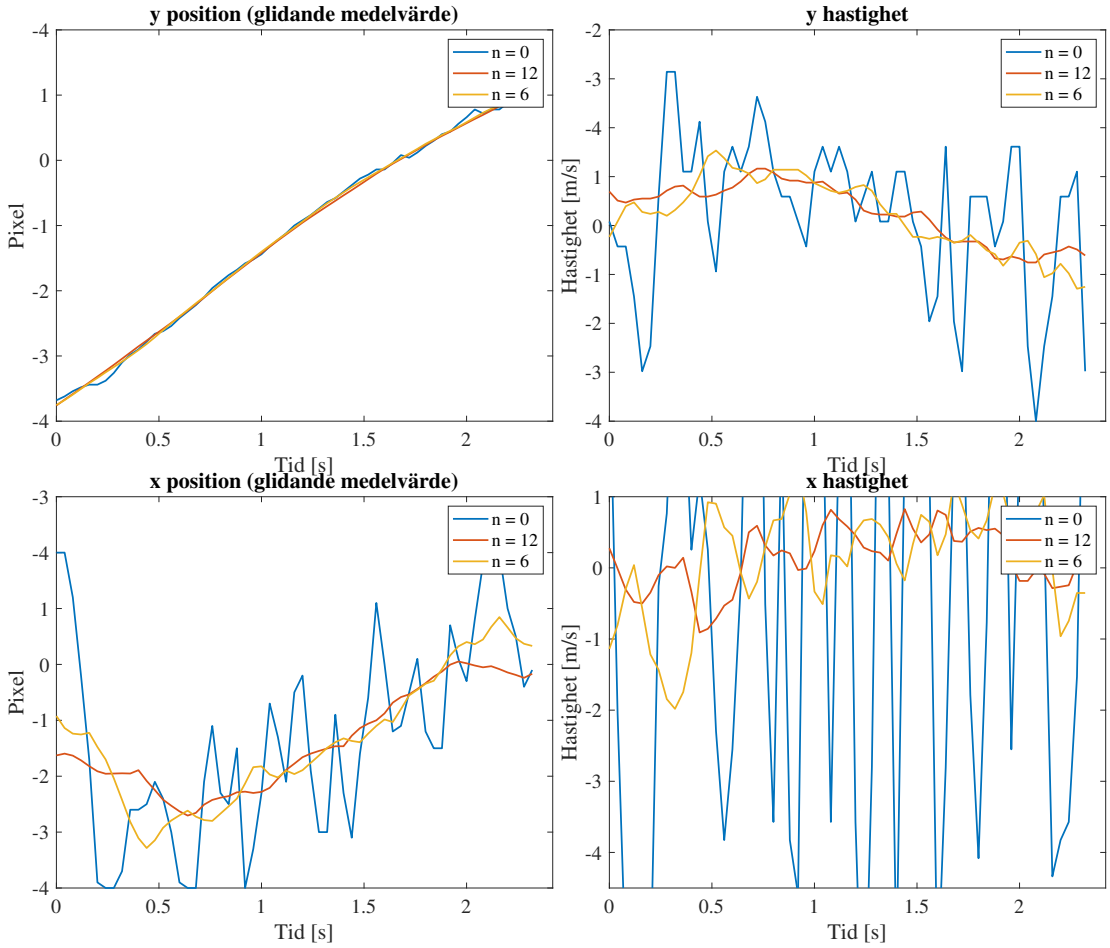
Figur A.7: En spelare som sprintar på en 5 m sträcka, definierad som Sprint (Mitten), tagen från en videofil. De olika färgade linjerna representerar de olika intervallen som valdes för det glidande medelvärdet.



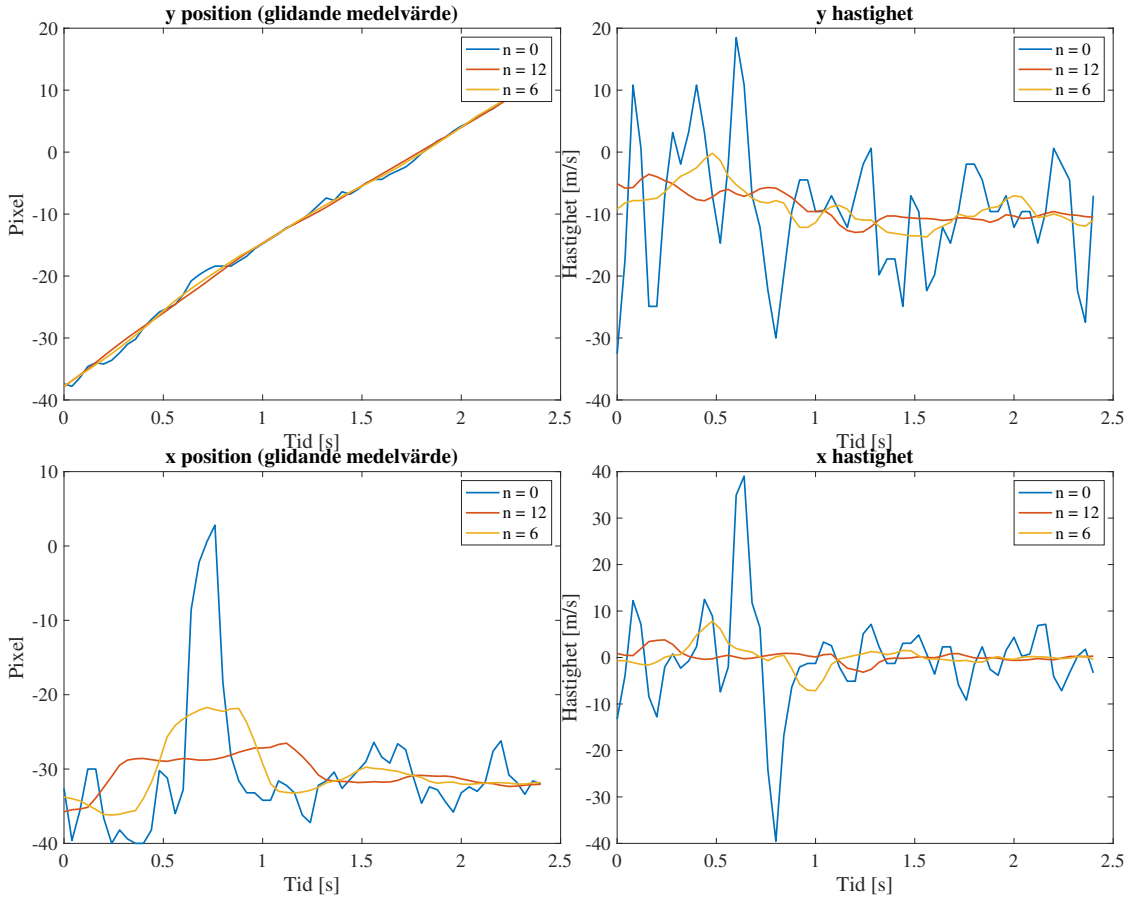
Figur A.8: En spelares rörelse på en 7,87 m sträcka, definierad som Diagonal, tagen från en videofil. De olika färgade linjerna representerar de olika intervallen som valdes för det glidande medelvärdet.



Figur A.9: En spelares rörelse på en 5 m sträcka, definierad som Längre, tagen från en videofil. De olika färgade linjerna representerar de olika intervallen som valdes för det glidande medelvärdet



Figur A.10: En spelares rörelse på en 5 m sträcka, definierad som Långt, tagen från en videofil. I detta fallet var målet bortplockat. De olika färgade linjerna representerar de olika intervallen som valdes för det glidande medelvärdet



Figur A.11: En spelares rörelse på en 5 m sträcka, definierad som Långt_mål, tagen från en videofil. Mätningarna gjordes för en spelare som blockerades av målet. De olika färgade linjerna representerar de olika intervallen som valdes för det glidande medelvärdet