STŘEDNÍ ŠKOLA PRŮMYSLOVÁ A HOTELOVÁ
UHERSKÉ HRADIŠTĚ

Středoškolská odborná činnost

Obor 18 - Informatika

Prostorové sledování objektů

Space object tracking

Vypracoval: **Aleš Křivák**

Škola: **Střední škola průmyslová a hotelová Uherské Hradiště**

Obor Technické Lyceum

Uherské Hradiště 2010

Čestné prohlášení

Prohlašuji, že jsem svoji práci vypracoval samostatně, pouze s využitím podkladů uvedených v použité literatuře a postup je v souladu se zákonem číslo 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů v platném znění.

V Uherském Hradišti 17. 3. 2010

Anotace

Aleš Křivák, Prostorové sledování objektů, Uherské Hradiště 2010

Tato práce se zabývá návrhem systému pro sledování pozice objektů v prostoru v reálném čase. Cílem je vytvoření funkčního algoritmu, který dokáže sledovat objekty pouze za pomocí dat z kamery, bez užití jiných polohovacích zařízení. Systémy užívající takovýchto čidel jsou totiž často drahé a nepřenosné.
V této práci není na rozdíl od konkurenčních systémů kladen přílišný důraz na přesnost, ale spíše na rychlost, robustnost a přenosnost.

Annotation

Ales Krivak, Space object tracking, Uherske Hradiste 2010

This project describes proposal of algorithm for real-time object tracking. The main goal is creation of algorithm to track objects just by camera, without using of special sensors. Systems working with this kind of sensors are mostly too expensive and impractical. This work is not focused on big accuracy, but its try to be quick, robust and portable.

Obsah:

1. Důvod vzniku projektu

Tento projekt vznikl jako pokus o vytvoření algoritmu pro sledování objektů v prostoru, bez nutnosti použití různých polohových čidel či jiného drahého vybavení.

V současnosti již existuje několik podobných prací, většina z nich je však pro počítač příliš náročná, a proto pracují pouze s nahraným videem – výpočty nejsou dostatečně rychlé, aby stíhaly pracovat s videem v reálném čase.

Cílem projektu je tedy vytvoření dostatečně rychlého algoritmu, který ke svému běhu potřebuje pouze data z kamery. Je také kladen důraz na robustnost a přenositelnost algoritmu, aby tak mohl být využit v mnoha situacích v praxi.

2. Části projektu

Pro systém jsou nezbytné pouze dvě části:

Počítač s libovolným operačním systémem podporujícím systém .NET a s připojením IE 1394, resp. USB 2.0.

Libovolná kamera s připojením přes USB nebo IE 1394, upevněná na stativu nebo na jiné stabilní podložce.

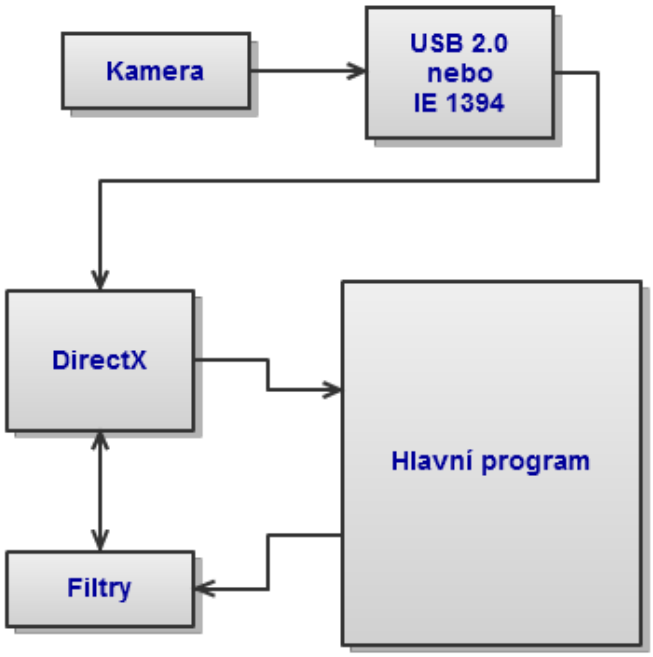
3. Popis algoritmu

3.1. Získávání dat z kamery

Získávání dat z kamery probíhá pomocí ovladačů DirectX. Toto rozhraní si přímo zjistí údaje, jako jsou typ kamery, rozlišení obrazu, barevná hloubka a jiné. Program pak nastaví filtr, který zachytává z datového proudu všechny nové snímky, ty pak uloží do paměti programu.

Rychlost procesu těsně závisí na snímkování kamery, to je ve většině případů 20 - 30 snímků za vteřinu. Program se totiž po každém zpracovaném snímku pozastaví, dokud filtr nezachytí nová data.

Celý proces načítání dat ukazuje obrázek 1.

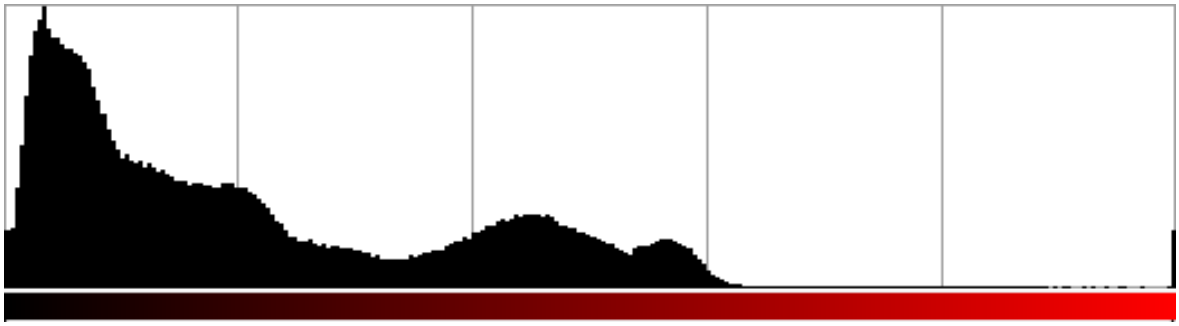


Obrázek 1: Připojení kamery

3.2. Histogramový filtr

Jednou z nejdůležitějších součástí projektu je tzv. histogramový filtr.

Pojem histogram pochází ze statistiky. Lze si jej představit jako graf výskytu určitých barev v obrázku (počtu pixelů dané barvy). Většinou se vytváří zvlášť pro každou barevnou složku daného obrázku.



Obrázek 2: Příklad histogramu pro červenou složku

V programu jsou histogramy použity k zaměření oblasti pixelů, reprezentujících daný objekt, a jejímu průběžnému sledování v zachytávaném videu. Vstupem do tohoto procesu je počáteční pozice objektu. Ta je zadána při startu programu a je použita k výpočtu histogramů pro červenou, zelenou, modrou a jasovou složku v blízkém okolí.

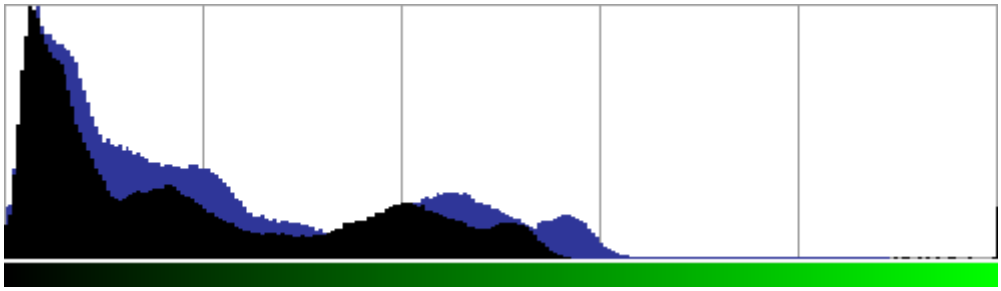
Při dalším pokračování algoritmu se pracuje s každým novým snímkem následovně:

1. Podle několika minulých snímků se určí pravděpodobná nová pozice.
2. Na novém snímku jsou spočítány histogramy pro červenou, zelenou, modrou a jasovou složku několika oblastí v okolí pravděpodobné pozice sledovaného objektu, získané z minulého kroku.
3. Všechny získané histogramy jsou porovnány s původními histogramy a je tak určen nejpravděpodobnější posun pozice objektu na aktuálním snímku. Postup při porovnávání histogramů je přesněji popsán v sekci 3.3.
4. Získaný posun je vyhlazen pomocí údajů z minulých snímků.

3.3. Porovnávání histogramů

Při porovnávání histogramů se pracuje vždy jen s jednou dvojicí histogramů. Nejprve jsou oba histogramy vyhlazeny. U každé barvy je spočítán vážený průměr s jejími sousedy. Tím se částečně odstraní šum.

Následně je v této dvojici každý výskyt barvy porovnán a je určena odchylka. Tato odchylka je následně umocněna, tím se opět zmenší náchylnost na malé poruchy v obraze, způsobené šumem.



Obrázek 3: Rozdíl histogramů – původní histogram černě, změna modře

Na závěr porovnávání jsou sečtené odchylky jednotlivých barev a jasu zprůměrovány a je tak určena výsledná chyba testované oblasti.

3.4. Hranový detektor

Pro další postup je kromě samotného určení nového umístění objektu potřeba taky mapa hran. V současné době existuje spousta různých algoritmů, pro výpočet hran v obrázku, většina je však náročná na výpočet, a proto nepoužitelná v případě realtimové aplikace.

V této práci je proto použit tzv. Sobelův filtr. Použitý algoritmus vypadá následovně:

1. Daná oblast je nejprve vyhlazena pomocí vyhlazovacího filtru.
2. Na oblast je použito vertikální a horizontální konvoluční jádro.

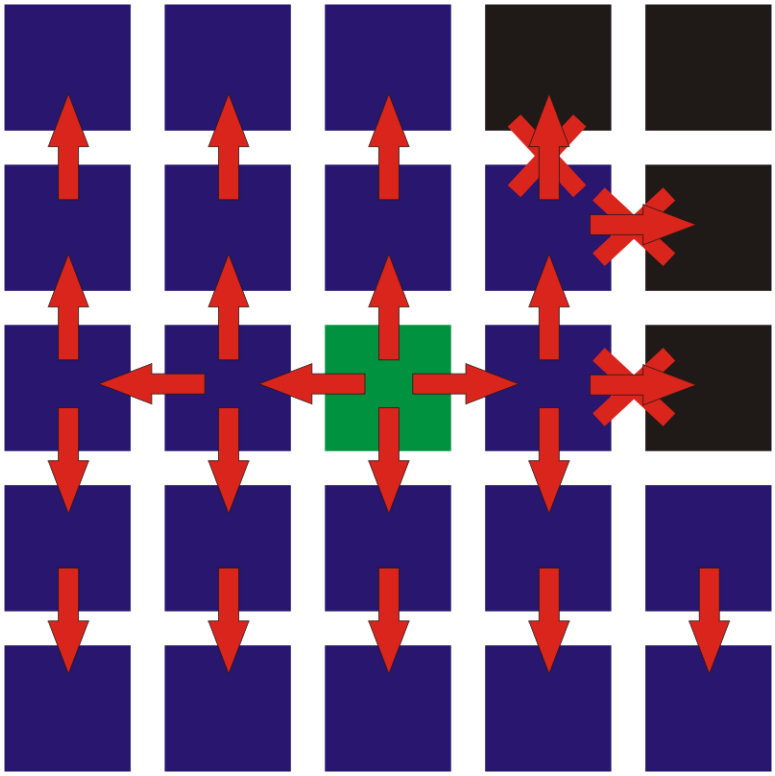
$$\mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad \mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$

3. Z výsledků konvoluce je vypočtena celková velikost $\mathbf{G} = \sqrt{\mathbf{G}_x^{\,2} + \mathbf{G}_y^{\,2}}$.
4. Na výsledky je použito prahování – podle velikosti **G** je určeno, zda se jedná o hranu či nikoliv.

3.5. Zaměření přesného středu objektu

K zaměření polohy objektu již byl použit histogramový filtr. Takto získaná poloha však není dostatečně přesná.

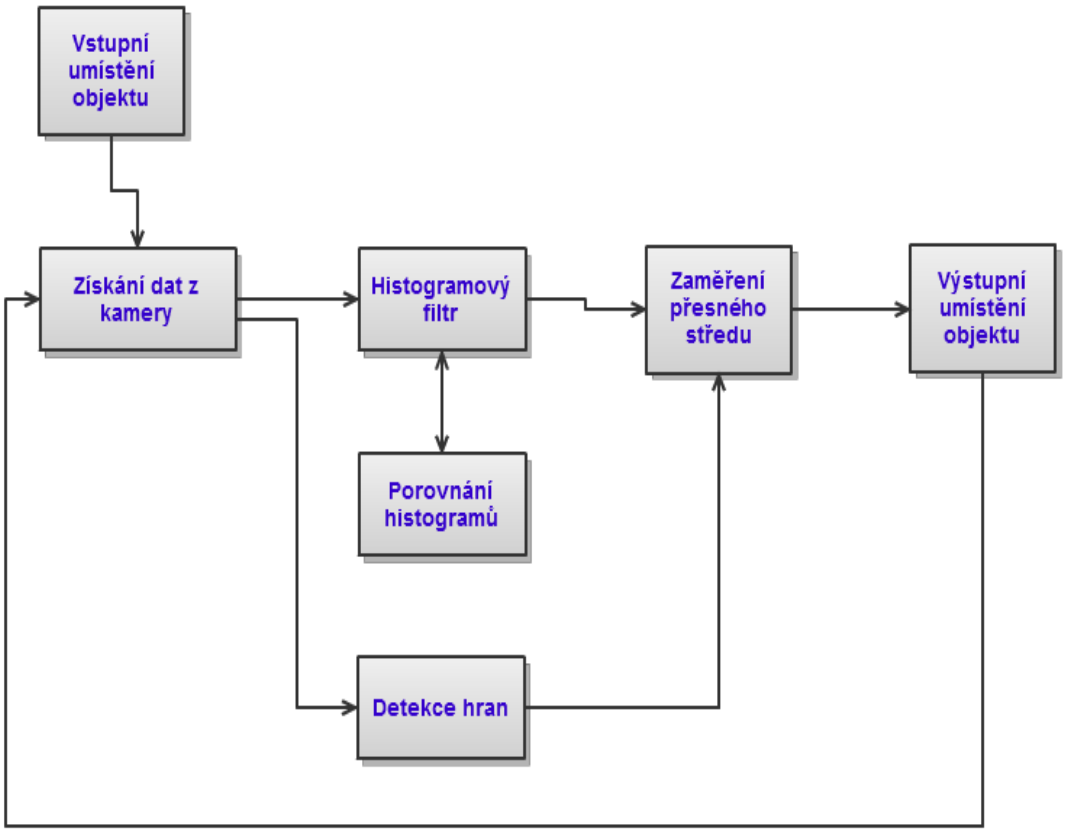
Pro přesné zaměření středu se provede určení barevné oblasti objektu. Vezme se několik pixelů umístěných v oblasti určené pomocí histogramového filtru.
Od těchto bodů se testují sousední body ve všech směrech a ověřuje se podobnost barvy. V případě, že se při procházení narazí na hranu určenou hranovou detekcí nebo je barva pixelu příliš odlišná, je testování v tomto směru přerušeno.



Obrázek 4: Schéma procházení pixelů – počátek zeleně, normální pixely modře, hranové pixely černě

Takto je získána celá oblast objektu a jako střed objektu je určen její střed.

3.6. Provázání jednotlivých částí algoritmu



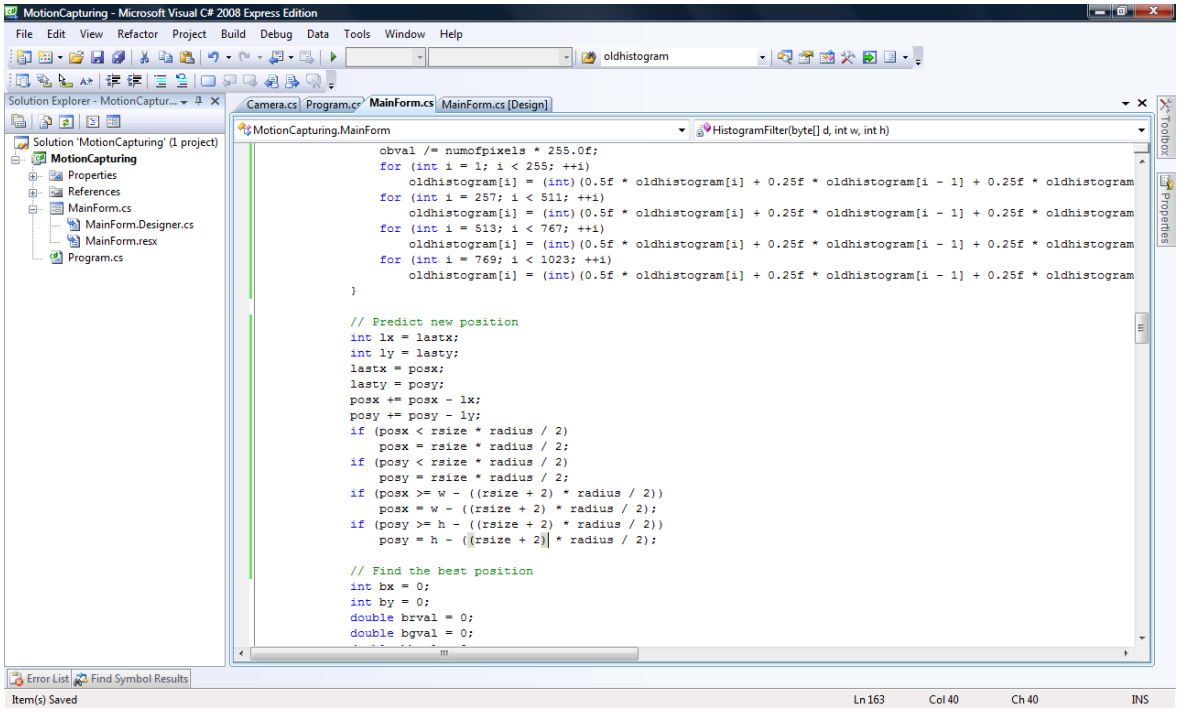
Obrázek 5: Schéma propojení algoritmu

4. Způsob tvorby programu

4.1. Vývojové prostředí

Celý systém je psaný v jazyce C#. K použití tohoto jazyka jsem se rozhodl, protože se jedná o svobodný jazyk, poskytující velkou volnost při tvorbě programů. Navíc je volně dostupné jeho vývojové prostředí Microsoft Visual C# Expres Edition, které poskytuje značné množství ladících nástrojů.

Většina jeho součástí je uložena v knihovnách a celé prostředí je tak snadno rozšiřitelné a také se snadno udržuje aktualizované.



Obrázek 6: Vývojové prostředí C#

4.2. Úvod do jazyka C#

C# byl vyvinut firmou Microsoft jako vlajková loď pro jeho nové běhové prostředí .NET. Je to plně objektově orientovaný jazyk (všechny objekty jsou potomky základní třídy, dokonce i základní typy jako například datový typ int).

Tento jazyk je také interpretovaný, to znamená, že při kompilaci se nevytváří přímo zdrojový kód pro procesor, ale tak zvaný byte code. Byte code je něco jako strojový kód pro virtuální procesor. Tento kód se překládá až při spuštění programu, to zajišťuje, že výsledný kód je přesně optimalizován na cílový procesor. Tento postup je velmi podobný jako u jazyku Java. Dokonce i Microsoft přiznává, že z velké části z tohoto jazyka čerpal.

Jak již bylo dříve řečeno, jazyk poskytuje výborné nástroje pro práci s knihovnami, proto je přímo žádoucí oddělení hlavní programové části od zbytku.

4.3. Systematické dělení kódu

Programy v jazyce C# jsou většinou rozděleny na několik logických částí, z nichž každá vykonává určitou část potřebnou pro běh programu. V ukázkovém programu je celý systém rozdělen v první řadě na hlavní program a sledovací algoritmus.

Tento postup umožňuje snadnou integraci sledovacího algoritmu do všech druhů aplikací, psaných v jazyce C#. Stačí přidat soubory s algoritmem do projektu, a pak již jen stačí jeho implementace do programu.

5. Závěr

Celá práce se snaží hlavně o ukázání možného řešení algoritmického problému a již neimplementuje jeho praktické využití.

I přesto se již v tomto stádiu nabízí několik případů užití. Jako příklad by mohla posloužit počítačová hra tenis. Projektor zobrazuje hru z perspektivy hráče a kamera, postavená vedle projektoru zaměřuje pozici rukou hráče. Tato pozice se přepočítá podle rychlosti a směru na úder tenisovou pálkou.

Jiným příkladem by mohla být kamera na dálnici, která sleduje směr a rychlost jízdy projíždějících aut.

Přestože samotný má algoritmus stále několik chyb, např. citlivost na změnu světelných podmínek, je již použitelný v praxi. Navíc hlavním cílem projektu je zlevnění a zpřístupnění tohoto postupu, oba dva úkoly se tedy podařilo splnit.

6. Poděkování

Děkuji všem, kteří mi pomohli při práci, jmenovitě svému bratru, Viktoru Křivákovi, za pomoc při hledání nových algoritmů a nezávislých pohledů. Také děkuji Ing. Bohumíru Brhelovi za silnou podporu.

7. Literatura

[1] *Wikipedia* [online]: www.wikipedia.org

[2] DONOSER Michael, BISCHOF Horst. *Fast Non-Rigid Object Boundary Tracking* [online]. [cit. 2010-03-10] Dostupné na www.comp.leeds.ac.uk/bmvc2008/proceedings/papers/58.pdf.

[3] KŘIVÁK Viktor. *Optická Střelnice.* Uherské Hradiště, Střední průmyslová škola, 2007.

8. Použitý software

[1] Microsoft Visual C# 2008 Express Edition (volně šiřitelné)

[2] Microsoft Office Word 2000 (Komerční program)

[3] GIMP 2.6.7 (GNU GPL)

[4] PDF Redirect v2 (freeware)

[5] Gliffy (zkušební verze)

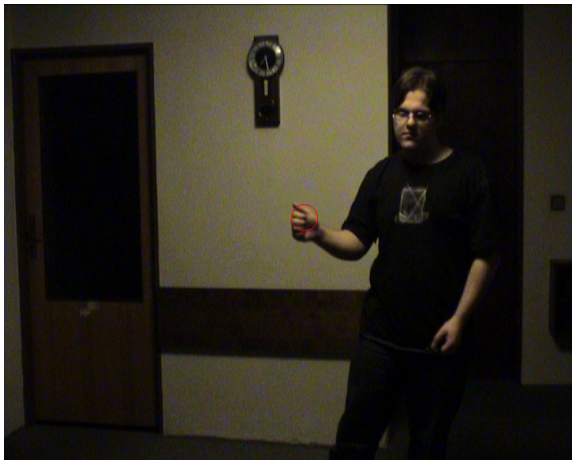
9. Seznam obrázků

Přílohy

A Ukázkové obrázky k jednotlivým krokům algoritmu

Příklad zaměření pozice pomocí histogramového filtru

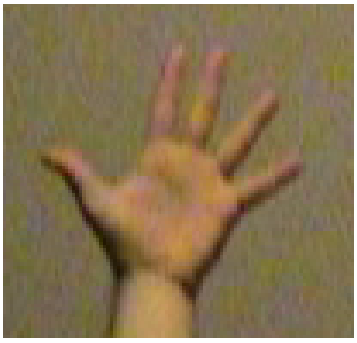
Snímek 1



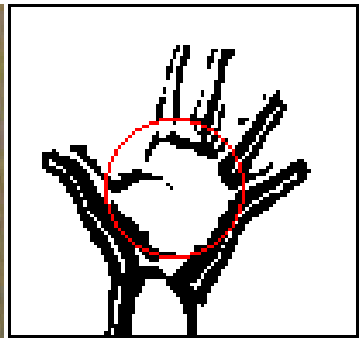
Snímek 20

Tyto snímky jsou od sebe posunuté o necelou jednu vteřinu. Lze na nich změnu polohy sledovaného objektu (v tomto případě ruky) i správné zaměření na obou snímcích (červené kolečko).

Příklad hranového detektoru



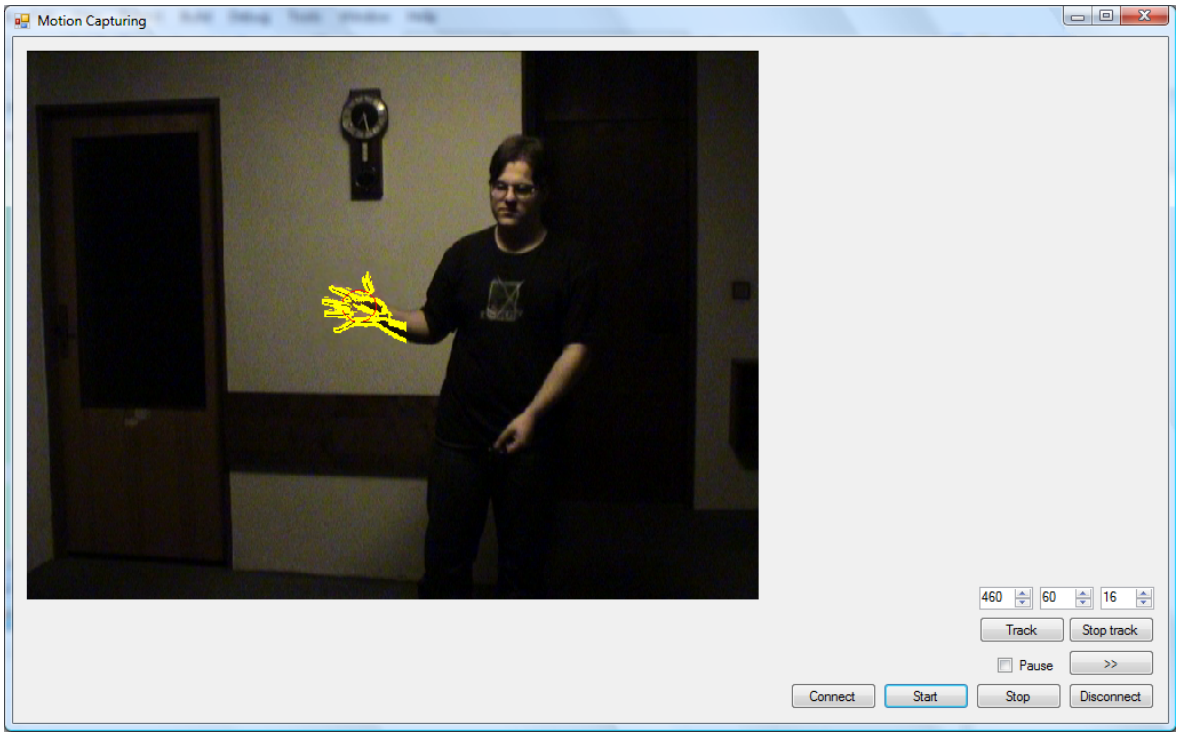
Zdrojová oblast



Mapa hran

Na obrázcích je vidět, jak vypadá původní zdrojová oblast i detekovaná mapa hran. Na mapě hran je také zaměřený střed objektu (červené kolo).

B Testovací program



Prostředí testovacího programu

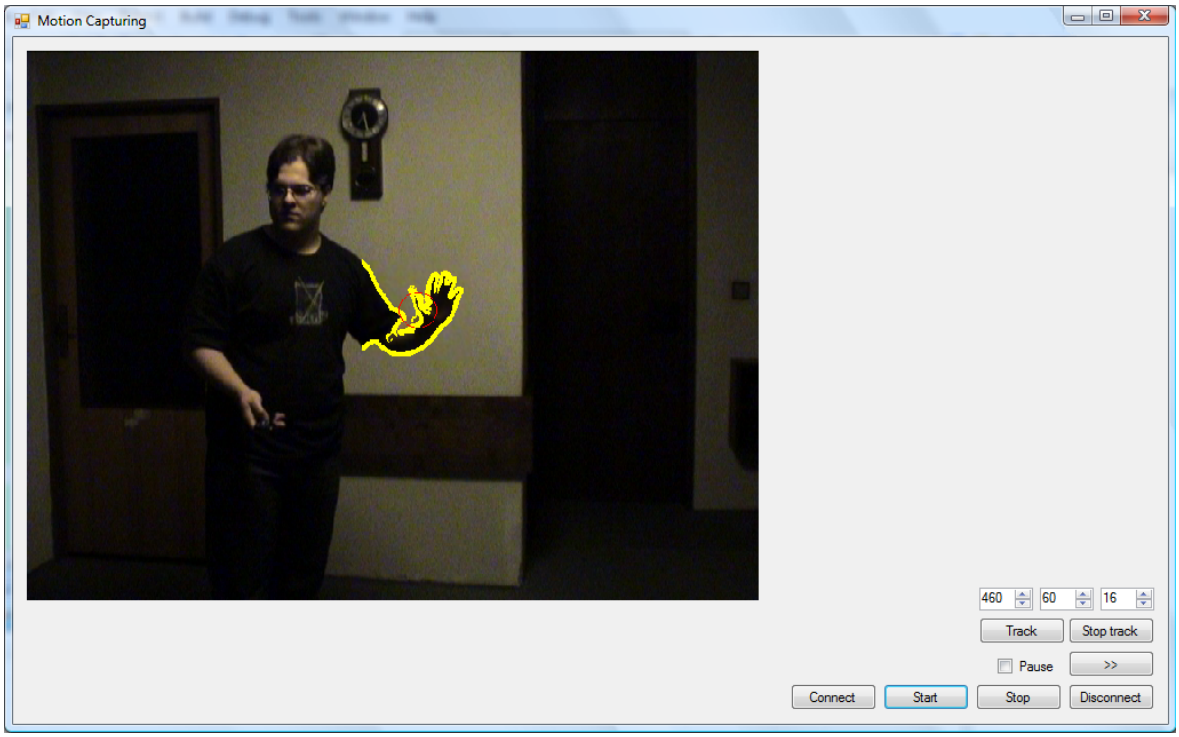
Ovládání:

Connect	Připojí rozhraní DirectX ke kameře
Start	Zahájí přenos dat z kamery
Stop	Ukončí přenos dat z kamery
Disconnect	Odpojí se od kamery
Pause	Pozastaví video
>>	Při pozastaveném videu umožňuje snímkování
Track	Zahájí sledování objektu na počáteční pozici, nastavené pomocí souřadnic x, y a poloměru sledovaného objektu
Stop track	Ukončí sledování objektu

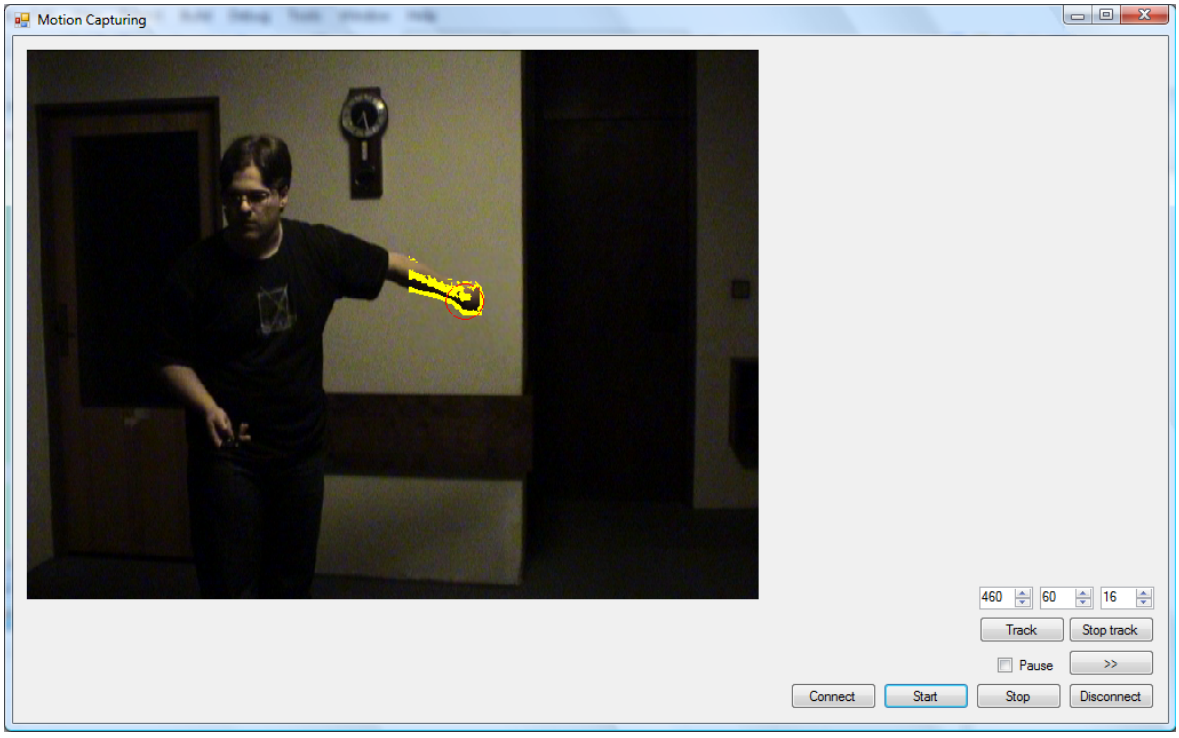
Výstup:

Testovací program implementuje celý zde popsaný algoritmus a zobrazuje jeho výstup jako aktuální pozici sledovaného objektu se zvýrazněnými hranami.

Ukázky programu při běhu:



Na tomto náhledu je dobře viditelný výsledek hranové detekce



Algoritmus se přizpůsobí i změně tvaru objektu (sevření ruky)