
Počítačové zpracování obrazu

Projekt Učíme se navzájem

Tomáš Pokorný, Vojtěch Přikryl
Jaroška • 14. prosince 2009



Obsah

Abstrakt	4
Začátky	5
M&M	5
Původní cíle	5
Programové vybavení	5
První kroky	6
Teorie	7
Snímání a zpracování obrazu	7
<i>Aktuální systém</i>	7
<i>Rozpracovaný systém</i>	7
Kalibrace	8
Transformace	8
<i>Poměrová transformace</i>	8
<i>Grafická transformace</i>	9
<i>Další transformace</i>	9
Zobrazení	9
Implementace	10
Snímání obrazu	10
Zpracování obrazu	9
Kalibrace	10
Transformace	10
Zobrazení	11
Aktuální stav	12
Aktuálně fungující	12
Rozpracované	12
Problémy	12

Budoucnost	13
Dokončení přepočtu	13
Možné aplikace	13
Poděkování	14

Abstrakt

Projekt Počítačové zpracování obrazu si klade za cíl pomocí kamery a projektoru připojeného k počítači umožnit uživateli, aby laserového ukazovátka mohl používat jako ukazatele na objekty zobrazené na projektoru. Aplikace zpracovávající obraz je napsána v Pythonu a využívá knihovny Python Image Library (PIL) pro analýzu a vykreslování obrazu. Funkce aplikace jsou tyto: kalibrace webkamery pro dané umístění, zachycení a načtení obrazu z webkamery, analýza a hledání nasvíceného bodu, transformace snímaného obrazu pomocí funkce a kalibračních dat a finální vykreslení nasvíceného bodu na projektoru. Rozpracováno je rychlejší snímání a zpracování obrazu a v budoucnu jsou plánovány aplikace a hry tento software využívající.

Začátky

M&M

Na soustředění korespondenčního semináře M&M jsem se s touto problematikou setkal v rámci tzv. konfery, což byla několika odpolední práce s následnou prezentací výsledků. Zde jsme dostali již předpřipravenou část softwaru pro snímání obrazu a pro promítání výsledků a dále jsme se zabývali zpracováním tohoto obrazu a jeho správným vykreslováním. Zde vznikly všechny dosud používané transformace a velká většina aktuálního kódu.

PŮVODNÍ CÍLE

Zadání znělo takto:

Cílem tématu je vybrat vhodnou detekovanou charakteristiku obrazu, vymyslet algoritmus na detekci vlastností v obrazu a pak naprogramovat a testovat na počítači s webkamerou, možná v kombinaci s projektorem.

Cíl je možné si vybrat, možnosti jsou např. detekce barevné tečky, sledování více teček či trajektorie pohybu tečky či jiného objektu. Bude potřeba z obrazu odstranit pozadí a perspektivní zkreslení. K tomu bude potřeba i geometrické počítání a transformace spolu s měřením či odhadem toho zkreslení. Pokud se povede dobrá kalibrace a odstranění zkreslení, může se promítat zpětná vazba přímo na snímání objekty.

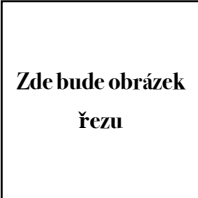
Původně jsme mysleli, že zpracování půjde snadno a že se za chvíli dostaneme k detekci pohybu a vykreslování čáry, ale ukázalo se, že transformace není vůbec jednoduchá a tak jsme u ní prakticky skončili, částečně vlivem její obtížnosti, částečně vlivem začátečnické chyby.

PROGRAMOVÉ VYBAVENÍ

Na M&M jsme pracovali na Linuxovém notebooku, ke kterému byla připojena webkamera a projektor, snímání probíhalo pomocí MPlayeru, ke zpracování sloužil Python s PIL a Tk pro zobrazování výsledků. Nynější vybavení je notebook s Mac OS X vybavený integrovanou webkamerou iSight, připojovaný k projektoru. Snímání je realizováno pomocí softwaru BTV. Za hlavní programovací jazyk stále slouží Python, ale pracuje se na využití jazyku Objective C v souvislosti s rychlejším načítáním snímků z kamery. Zpracování stále probíhá pomocí PIL a zobrazování pomocí Tk.

PRVNÍ KROKY

Ze začátku jsme se učili pracovat s Pythonem, protože jsme ho nikdo pořádně neuměli, a poté jsme přistoupili k vlastnímu zpracování obrazu. Zjišťovali jsme, jak obrázek vypadá, co se v něm všechno dá najít, jak je charakteristický nasvícený bod a zkoušeli jsme ho hledat. Poté přišla na řadu transformace obrazu z kamery, na které jsme strávili dlouhý čas. Vzniklo několik konkurenčních algoritmů, avšak žádný nefungoval nijak zvlášť přesvědčivě. Nakonec se nám podařilo zprovoznit dvě transformace s relativně dobrými výsledky a úspěšně zprovoznit vykreslování.



**Zde bude obrázek
řezu**

Teorie

SNÍMÁNÍ A ZPRACOVÁNÍ OBRAZU

Aktuální systém

Obraz je snímán pomocí programu BTV, který umí ukládat jednotlivé snímky z kamery. Program je nastaven tak, aby snímal snímek každé 0,3 sekundy a ukládal jej do dané složky, odkud je čte skript. Skript si na začátku zjistí, jaké soubory daný adresář obsahuje a pokud se objeví nový soubor, načte ho a dále ho zpracovává. Zpracování probíhá tak, že načtený soubor je převeden na pole pixelů, které se postupně prochází a vyhodnocuje. Jsou 2 stupně kontroly:

1. První stupeň odfiltrovává hned při načítání pixely, jejichž barva se ani vzdáleně nepodobá barvě nasvíceného bodu a tyto pixely zahazuje, šetří tím výkon na další zpracování. Filtrování funguje na základě barvy, hledaný nasvícený bod má určité předem zjištěné spektrum, a pokud bod právě načtený nemá dané spektrum a nepohybuje se ani v okolí tohoto spektra, je vyhodnocen jako neplatný a dále se s ním nepočítá.
2. Druhý stupeň filtrování je blokový. U bodů prošlých minulým sítím se program podívá na jejich sousedy a spočítá součet jednotlivých barevných složek této plochy. Pokud poměr součtů jednotlivých barev není v daném poměru, je opět tento blok zahozen a dále se s ním nepočítá.

Pokud blok projde oběma kontrolami, je zjištěno maximum hodnoty součtu hledané barevné složky a definitivně je vybrán ten bod, který má tento součet nejvyšší. Tato metoda se ukázala být velmi spolehlivá, ale je potřeba najít vhodné nastavení konstant pro zahazování bodů, protože obzvláště při denním osvětlení či osvětlení různými zdroji světla jsou snímány body různých barev a je potřeba co nejvíce špatných zahodit, ale ne zahodit s nimi i ty, co chceme najít.

Rozpracovaný systém

Nyní pracuji na systému, který by získával obrázek nikoli přes soubor v adresáři, což je pomalé a náročné na výkon, ale přímo přes funkce operačního systému. Tato funkce je již závislá na operačním systému, zde konkrétně používám frameworky pro Mac OS X. Další funkce filtrování obrazu jsou již stejné jako u minulého způsobu.

KALIBRACE

Kalibrace je prvním s dějů souvisejících s transformací. Cílem kalibrace je zjistit, v jakých bodech na snímaném obraze se zobrazují rohy obrazu zobrazovaného dataprojektorem. To je důležité proto, abychom dokázali přepočítat bod nalezený na obraze webkamery na bod, který zobrazíme na projektoru. Nyní k funkci kalibrace. Program postupně rozsvítí 4 čtverečky v rozích obrazu projektoru a vždy chvíli počká, aby se kamera přizpůsobila, pak je sejme již popsaným způsobem a najde je. Tímto získáváme 4 kalibrační body, se kterými již můžeme počítat v transformacích.

TRANSFORMACE

Toto je nejdůležitější a matematicky nejobtížnější část celého projektu, která zajišťuje správné zobrazení bodu na projektoru. Co to je a proč je potřeba?

Předpokládejme, že máme projektor, který promítá na plátno obdélníkový obraz. Pokud by ho promítal zkreslený, má své vlastní funkce ke korekci tohoto zkreslení. Pokud nyní obraz snímáme kamerou, získáme nějaký útvar. Optimální situace nastává, pokud kamera snímá stejného bodu, jako promítá projektor. V tom případě je snímáný útvar obdélník se stejným poměrem stran k promítanému obrazu. Tady nám stačí úplně základní transformace a to jednoduché přínásobení konstantou odpovídající poměru délky strany snímáného a promítaného obrazu. Toto je ale situace, která nastane velmi zřídka. Většinou snímáme kamerou obraz zkreslený, protože pokud si představíme to, co snímá kamera, jde o jakýsi nekonečně vysoký čtyřboký jehlan. Pokud ovšem snímáme útvar mimo osu projektoru, bude snímáný útvar určitým řezem daného jehlanu, tudíž to bude docela obecný čtyřúhelník. My potřebujeme tento čtyřúhelník roztáhnout vhodně tak, abychom získali obdélník, který bude odpovídat obdélníku promítanému projektozem, abychom mohli tento obdélník zobrazit. To je právě záležitost transformace

Poměrová transformace

Toto je nejlepší námi naprogramovaná transformace. Základem je fakt, že i ve zkresleném útvaru zůstanou zachovány poměry vzdáleností a to speciálně poměrů vzdáleností nalezeného osvětleného bodu od stran. Pokud tuto úvahu zobecníme na dva rozměry, získáme rovnice, pomocí kterých můžeme získané souřadnice nalezeného bodu přepočítat na souřadnice bodu na obrazovce.

Zde bude obrázek řezu

Grafická transformace

Tato transformace dává nejlepší výsledky. Je postavena na funkci z knihovny PIL, která umí převést zadaný čtyřúhelník na obdélník. To je přesně to, co potřebujeme, ovšem má to jednu poměrně zásadní nevýhodu. Protože je to funkce z grafické knihovny, znamená to, že tuto transformaci umí provést pouze s obrázkem, nikoli jen daty. Je tedy potřeba vytvořit obrázek, zakreslit do něj nalezený bod, obrázek transformovat a opět transformovaný bod nalézt. To jsou hlavní 2 důvody, proč je transformace pomalá, což je její hlavní nevýhoda. Musí provést transformaci bitmapy, tzn. přepočítat každý pixel a pak my ji ještě musíme projít, abychom transformovaný pixel našli.

Další transformace

Během vývoje vznikly ještě 2 další transformace a to transformace přibližná, která vždy předpokládala, že útvar je lichoběžník a to postupně v obou rozměrech a výsledek pak zprůměrovala. Toto byla nejúspěšnější transformace z námi naprogramovaných, než jsme přišli na chybu v transformaci poměrové a nebo než jsme zvětšili úhel mezi osou promítaného obrazu a kamerou.

Druhá ze vzniklých transformací byla založena na úhlech a postupném posouvání kalibračních bodů do vrcholů promítaného čtyřúhelníka. Při nasimulování její činnosti ale bylo zjištěno, že byly úvahy chybné a tato transformace nemůže fungovat.

Ověřeno 15.12. úvahou a
Metapostem

ZOBRAZENÍ

Poté, co je nalezen bod, který se má vykreslit na projektoru, ho již stačí jenom vykreslit. Obraz se vykresluje do černého okna o velikosti obrazu promítaného projektorem. Tato část nepotřebuje nijaké zvláštní komentáře, jediné, na co je třeba si dávat pozor, je, abychom křížek v místě vykreslovaného bodu vykreslovali správnou barvou, kterou nevyhodnotí program jak nasvícený bod ukazovátkem.

Implementace

SNÍMÁNÍ OBRAZU

Původní systém byl přes MPlayer, jehož příkaz `mplayer tv:// -tv driver=v4l2:device=/dev/video:width=320:height=240:noaudio -vo jpeg` zajistil, aby se ukládaly z kamery obrázky ve formátu JPEG do zadané složky. MPlayer na Mac OSu si ale nedokázal s integrovanou webkamerou poradit a tak přišel na řadu program BTV, který dělá v tomto případě to stejné, co MPlayer, akorát s 5x vyšší náročností na výkon. Ze složky jsou funkcemi modulu `os` načítány, poté rozloženy na pole pixelů pomocí funkce `open()` a `load()` z PIL.

ZPRACOVÁNÍ OBRAZU

Jednotlivé pixely procházeny ve for cyklu procházeny a prochází zkouškou, zda mají vyhovující barvu. Pokud neprojdou, jsou začerněny. Pokud projdou, je spočítán opět for cyklem čtvereček o straně 6pixelů okolo daného bodu a v něm jsou složky sečteny a porovnává se poměr chtěné složky k nechtěné. Ten v současné době je nastaven na 1,3. Pokud projde pixel i tímto testem, zjistí se, zda je větší než maximum a pokud ano, tak je prohlášen za nové maximum.

KALIBRACE

Kalibrace funguje tak, že program vždy zobrazí bod v rohu obdélníka promítaného projektořem, poté chvíli počká, než si kamera přivykne a pak prohledá získaný obraz, zda se v něm nachází daný bod. Takto to udělá pro všechny čtyři body po směru hodinových ručiček a na konci zobrazí kalibrační body v kontrolním okně aplikace a uloží si je do pole, odkud jsou načítány transformačními funkcemi.

TRANSFORMACE

Poměrová transformace

Tato transformace je počítána funkcí `karltrans(x,y)`, kde (x,y) je dvojice souřadnic nalezeného bodu v útvaru a která obsahuje potřebné výpočty pro dané kalibrační body a nalezený bod v útvaru.

Grafická transformace

Tato transformace je realizována voláním knihovní funkce `transform((x, y), Image.QUAD, dd)`, kde (x,y) je dvojice rozměrů výstupního obrázku a dd je pole vrcholů nalezeného útvaru

ZOBRAZENÍ

Zobrazení je jednoduchá funkce, která vezme transformované souřadnice a na ně nakreslí dvě čáry tak, aby se protnuly v místě vypočteného bodu. Kreslí se na černé okno bez okrajů, aby nevznikala špatná rozpoznání osvětleného bodu.

Aktuální stav

AKTUÁLNĚ FUNGUJÍCÍ

Aktuálně funguje zachytávání obrazu pomocí BTV, jeho zpracování, z transformací jsou využívány transformace poměrová a transformace grafická, k vykreslování se používá okno využívající Tk z původního projektu.

ROZPRACOVANÉ

Aktuálně rozpracovaný je přechod na nové získávání snímků z webkamery, rovněž probíhají první pokusy s přepisováním do Objective C. Co se týče transformací, budu se dále snažit vylepšit a zjednodušit poměrovou transformaci.

PROBLÉMY

Prvním, zásadním a nejhloupějším problémem byl problém s poměrovou transformací. Jeho podstata byla v odlišném číslování rohů obdélníka při kalibraci a při transformaci. Tato chyba nás zdržela o jedno odpoledne, kdy jsme mohli pokračovat v programování. Byla odhalena až asi v polovině prezentace konfery na soustředění M&M.

V současné době se potýkám s velkým problémem při pokusu o nový způsob získávání obrázků. Při získávání nového obrázku je na něm provedeno několik převodů a ač by měly být při načtení dalšího obrázku výsledky těchto převodů přepsány, neděje se tak a tudíž se RAM plní nevyužitými předzpracovanými obrázky a nejde se jich nijak zbavit. Tento problém se vyskytuje pouze v Pythonu a ne v Objective C a je to hlavní hnací motor přechodu na Objective C.

Budoucnost

DOKONČENÍ PŘEPOČTU

Nejdůležitějším cílem do budoucnosti je dokončit přechod na nový způsob získávání snímků z kamery bez ukládání do adresáře, což je jeden pilíř, který by měl urychlit zpracování a zrychlit transformaci obrazu, aby byla rychlá v 320x240 a použitelná v rozlišení VGA. Tím by se stalo zpracování použitelné pro další aplikace, vyžadující rychlejší odezvu.

PŘEPSÁNÍ DO OBJECTIVE-C

Po dokončení a doladění transformací bude zahájen pokus přepsat stávající kód do jazyka Objective C s využitím knihoven a frameworků Mac OSu. Tím by se měl jednak zrychlit výpočet, jednak odstranit některé problémy způsobené použitím rozhraní Tk a možná to bude nutná podmínka pro nový způsob získávání snímků, alespoň část kódu přepsat do Objective C.

MOŽNÉ APLIKACE

Kreslení

Místo toho, aby se zobrazoval jen křížek pod nasvíceným bodem bude laser za sebou zanechávat barevnou stopu. Tímto způsobem bude možné kreslit jednoduché obrázky a může být i možnost změny barvy. Opět je ale potřeba, aby barva čáry kreslené laserem nebyla stejná jako nasvícený bod, aby nebyla chybně zachycena jeho poloha.

Střílení balonků

Jednoduchá hra pro 1, v budoucnosti 2 hráče s různě barevnými laserovými ukazovátky, kteří budou pomocí svícení na balonky zobrazené na projektoru na tyto balonky střílet a získávat za ně body.

Ovládání myši

Pomocí laserového ukazovátka bude možné ovládat kurzor myši. Zde je problém s barvou oken, která musí být diametrálně odlišná od barvy ukazovátka, aby nedocházelo k chybným detekcím. Rovněž by bylo potřeba detekovat kliknutí, například pomocí bliknutí laserem.

Odkazy a zdroje

Seminář M&M: <http://mam.mff.cuni.cz/>

Referenční příručka Python Image Library

Mac OS X Reference Library

Zdrojové kódy programu a jeho stránky: <http://code.google.com/p/zpracovaniobrazu/>

Poděkování

Tomáši Gavenčakovi za vedení konference na M&M

Karlu Královi a Lukáši za vymyšlení poměrové transformace

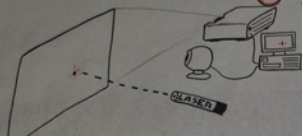
Petrovi za spolupráci na konferenci

Mgr. Marku Blahovi za technické zázemí a podporu při pokračování projektu

A všem dalším, kteří mě svými radami nasměřovali na správnou cestu

Poster z M&M

ZPRACOVÁNÍ OBRAZU



Užite vybavení:

- webkamera
- zelený laserové ukazovátko
- dataprojektor
- PC - Python + PIL

Rozpoznání barevného bodu

I. Počáteční filtr:

- 1) Zjištění spektra hledaného bodu
 - zelený laser - vysoký jas
 - nízký podíl červené barvy
- 2) Odfiltrování nevýhrujících bodů
 - jas zelené < 60%
 - jas červené > 60%

II. Skenování čtverečkem:

- bod osvětlený laserem se zobrazí jako barevný čtvereček o straně 8/6px
- čtverečkem projdeme body zbylé po I.
- vybereme ten, který:
 - má vhodný poměr zelené a červené barvy
 - má nejvyšší celkový jas zelené složky
- za hledaný bod prohlásíme jeho střed

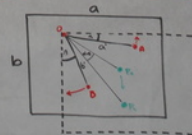
Transformace obrazu

I. Poměrová transformace



- vychází ze zachování poměrů při zkrácení

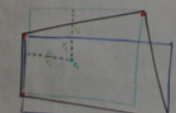
II. Úhlová transformace



$10R/10R = \frac{a}{b} \cdot \frac{b}{a}$

- vychází z posunutí kalibračních bodů do jejich skutečné polohy a odpovídajícího posunutí bodu P_i

III. Průměrová transformace



- souřadnice bodu P_i jsou $\left[\frac{x_i + x_j}{2}, \frac{y_i + y_j}{2} \right]$

IV. Grafická transformace pomocí knihovny

- čtýřúhelník s nalezeným bodem transformujeme pomocí knihovní fce
- v transformovaném obrázku najdeme hledaný bod