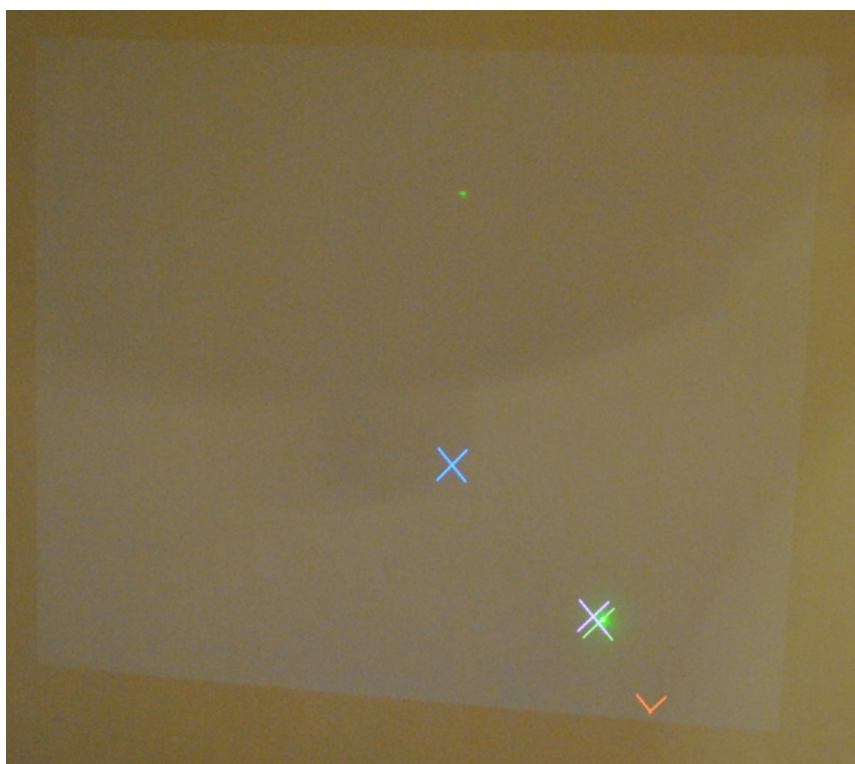

Počítačové zpracování obrazu

Projekt Učíme se navzájem

Tomáš Pokorný, Vojtěch Přikryl
Jaroška • 23. března 2010



Obsah

Abstrakt	4
Začátky	5
M&M	5
Původní cíle	5
Programové vybavení	5
První kroky	6
Teorie	7
Snímání a zpracování obrazu	7
Kalibrace	7
Transformace	7
<i>Poměrová transformace</i>	8
<i>Transformace se 2 body</i>	8
<i>Grafická transformace</i>	9
<i>Další transformace</i>	9
Zobrazení	10
Implementace	11
Snímání obrazu	11
Kalibrace	11
Transformace	12
<i>Poměrová transformace</i>	12
<i>Transformace se dvěma body</i>	12
<i>Grafická transformace</i>	12
Zobrazení	12
Aktuální stav	13
Aktuálně fungující	13
Rozpracované	13

Problémy	13
Budoucnost	14
Dokončení transformace	14
Možné aplikace	14
<i>Kreslení</i>	14
<i>Střílení balonků</i>	14
<i>Ovládání myši</i>	14
Odkazy a zdroje	15
Poděkování	15
Poster z M&M	16

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 je napsána v Objective C. Funkce aplikace jsou tyto: kalibrace webkamery pro dané umístění, zachycení a načtení obrazu z webkamery, analýza a hledání nasvíceného bodu, transformace snímaného obrazu pomocí transformační funkce a kalibračních dat a finální vykreslení nasvíceného bodu na projektoru. Rozpracováno je zpřesnění transformace 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 snímané 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 frameworku CocoaSequenceGrabber v programu HledaniBodu. Tento program je napsán v Objective C a v něm rovněž probíhá hledání nejsvětějšího bodu, transformace jeho souřadnic a jeho vykreslení na projektor.

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.

Teorie

SNÍMÁNÍ A ZPRACOVÁNÍ OBRAZU

Obraz je snímán pomocí frameworku CocoaSequenceGrabber v programu HledaniBodu. Program snímá obrázky tak, jak jsou poskytovány kamerou a na hledá v nich nejsvětlejší bod a transformuje jeho souřadnice. Hledání probíhá tak, že je obrázek převeden na pole bytů a nad ním se vždy počítá součet 5x5 pixelů po jednotlivých barevných složkách. Před tímto výpočtem můžou být nastaveny omezující podmínky, kdy je zbytečné výpočet provádět, protože jsou body příliš tmavé než aby na ně bylo svíceno.

Z jednotlivých blokových součtů je zjištěno maximum hodnoty součtu hledané barevné složky a definitivně je vybrán ten bod, který má tento součet nejvyšší. U tohoto maxima jsou ještě kontrolovány součty jednotlivých složek, aby se omezily artefakty kdy pokud na projektor nesvítím se nezachytávaly zobrazené křížky.

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 nezahodit s nimi i ty, co chceme najít.

KALIBRACE

Kalibrace je prvním z 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 popsáním způsobem a najde je. Tímto získáváme 4 kalibrační body, se kterými již můžeme počítat v transformacích.

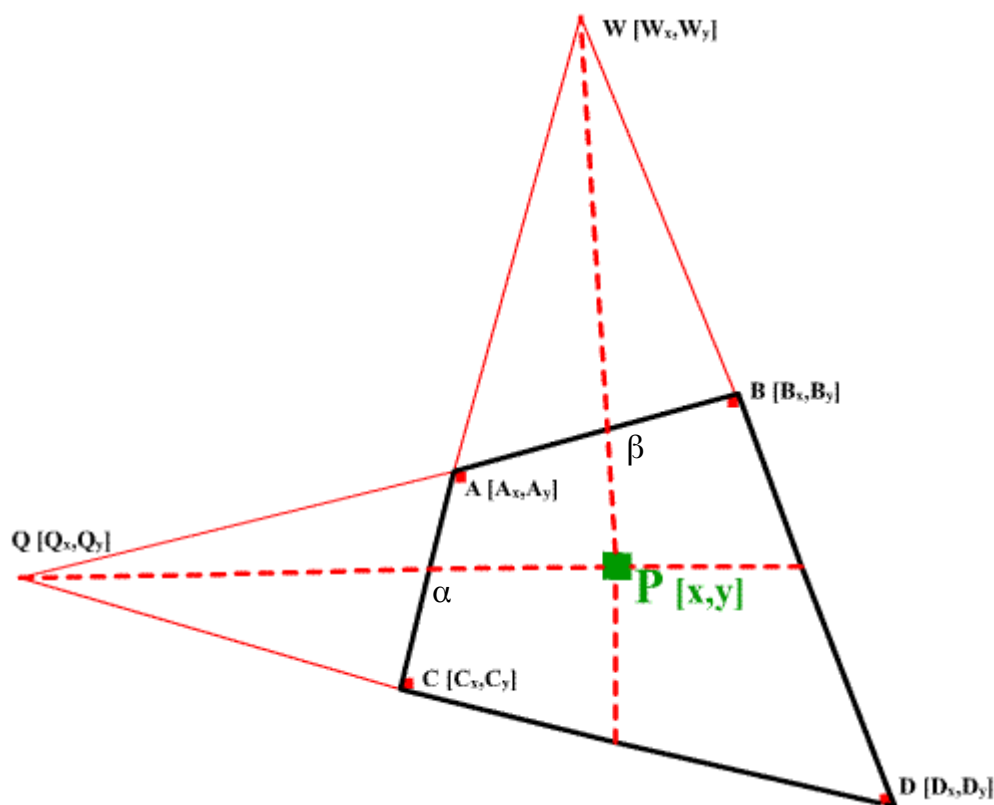
TRANSFORMACE

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 ze 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 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 projektorem, abychom mohli tento obdélník zobrazit. To je právě záležitost transformace

Transformace se 2 body



Toto je nová transformace vymyšlená Lukášem Langerem a je v současnosti používána. Základem je to, že při zachycení zkresleného obrazu se původně rovnoběžné přímky určené rohovými body A, B, C, D budou sbíhat a tudíž se protnou v nějakém bodě (Q, W). Tento bod jednoduše nalezneme prodloužením stran. Tímto bodem prochází i přímka určená

zachyceným bodem P a rovnoběžná se stranou. Přímka procházející zachyceným bodem P a rovnoběžná se stranou AC nebo AB bude při zkreslení procházet nalezeným průsečíkem. Nalezneme průsečík této přímky se stranou AB nebo AC, nazveme ho α a β a určíme poměr části k průsečíku k celé této straně. Toto provedeme pro obě rovnoběžné strany a získané poměry stačí vynásobit šířkou a výškou zobrazovaného obrazu. Tato transformace dává nejlepší výsledky s přesností okolo 5cm.

Poměrová transformace

Toto byla 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.

Grafická transformace

Tato transformace se uplatnila hlavně v začátcích. 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. Navíc je tato transformace závislá na Pythonu, při přepsání transformace do Objective C musela být tato transformace vypuštěna.

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 používaná transformace v začátcích, než jsme přišli na chybu v transformaci poměrové. Se stoupajícím zkreslením rapidně klesá přesnost.

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.

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 nasvětlený bod ukazovátkem.

Implementace

Celý proces a vzájemná volání metod jsou zakreslená v grafu v souboru `schema_kompletni.vue` a pdf.

Tato část bude přepsána jako dokumentace jednotlivých metod...

SNÍMÁNÍ OBRAZU

CocoaSequenceGrabber

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 dělal v tomto případě to stejné, co MPlayer, akorát s 5x vyšší náročností na výkon. Nakonec jsem se začal věnovat Objective C a naprogramoval program HledaniBodu, který pomocí frameworku CocoaSequenceGrabber zachytává přímo snímky z kamery.

ZPRACOVÁNÍ OBRAZU

ZOProcessImage

Obraz je převeden na pole bytů, z nichž každý udává jednu složku barvy pixelu. Program proto skáče po 4 prvcích pole (složky obrazu jsou “RGBA”) a pro každý pixel spočítá pomocí metody `getSumSquareAtIndex` součet jednotlivých složek ve čtverečku 5x5. Každý pixel je ověřován, jestli má dostatečný jas a pokud nemá, sumační čtverec se nepočítá. Filtrování je i na výstupu “čtverečkovací” metody, určuje minimální jas nejjasnějšího sumačního čtverce, nastavuje se pomocí posuvníků v GUI.

KALIBRACE

ZOCalibrate

Kalibrace funguje tak, že je zobrazena černá obrazovka a spuštěn časovač. Poté, co doběhne je zobrazen bod v rohu obrazu promítaného projektoem a spuštěn druhý časovač. Po jeho doběhu je opět zobrazena černá obrazovka a navíc nalezen nejsvětlejší bod minulého obrazu,

Takto je to pro všechny 4 body. Poté je zavolána metoda `initWithCalibrationArray:(NSArray *)`, kde `NSArray *` je pole s kalibračními body. tato metoda připraví pomocné kalibrační konstanty a tím je vše připraveno k transformaci.

TRANSFORMACE

Poměrová transformace

`ZOTransform`

Tato transformace je počítána metodou `transformPoint:(NSPoint)` z třídy `ZOTransform`, kde `NSPoint` je struktura obsahující dvojici souřadnic nalezeného bodu v útvaru. Třída obsahuje potřebné výpočty pro dané kalibrační body a nalezený bod v útvaru.

Transformace se dvěma body

`ZO2PointTransform`

Tato transformace, reprezentována třídou `ZO2PointTransform` má stejné rozhraní jako transformace poměrová, tedy transformace se počítá metodou `transformPoint:(NSPoint)`. Třída opět obsahuje potřebné výpočty a konstanty nutné pro přepočet.

ZOBRAZENÍ

`ZO2ProjectorView`

Zobrazení je řešeno třídou `ZOProjectorView`, která ovládá `NSView`, na kterém jsou body zobrazovány. Bod se určuje metodou `setPoint1:(NSPoint)` a `setPoint2:(NSPoint)`. Tato třída také obstarává zobrazování kalibračního obrazce při kalibraci, a to pomocí metody `setCalPoint:(int)`, kde `o` je normální běh a `1` až `4` jsou čísla kalibračních bodů.

Aktuální stav

AKTUÁLNĚ FUNGUJÍCÍ

Aktuálně funguje celý základní proces - načtení, zpracování, transformace i zobrazení. Fungují v základní podobě, vše je potřeba nastavovat ve zdrojovém kódu.

ROZPRACOVANÉ

Rozpracovaná je první aplikace, a to kreslení laserovým ukazovátkem. Je ve velmi raném stádiu vývoje, je velmi náročná na výpočetní výkon při větším množství objektů. Rovněž je téměř u konce rozdělení programu do tříd na jednotlivé úkony. V nejbližší době se chystám zprovoznit nastavování parametrů přes GUI.

PROBLÉMY

Prvním, zásadním a nejloupějším 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.

Problémy byly při přepisování do Objective C, ale byly úspěšně překonány. Největším problémem bylo vždy rozbíhat danou transformaci, protože při přepisování do počítače vznikla vždy alespoň jedna chyba, která se dost těžko hledala. Tyto problémy se táhly dlouhou dobu, je to vidět i na tom, že první námi naprogramovaná transformace se podařila zprovoznit až za dlouhou dobu a například zprovoznit dvoubodovou transformaci trvalo od 27. ledna, kdy byla uveřejněna na SVN až do 18. března, kdy vyšla stabilní revize.

Budoucnost

GUI, VGA

Nejdůležitějším cílem je dokončit rozdělení do mnoha tříd, tyto třídy zdokumentovat a vyčistit od balastu vzniklého při programování. Následně je v plánu umožnit nastavování všech potřebných parametrů zpracovávání obrazu přes GUI. Poté bude možný přechod na rozlišení VGA, které umožní přesnější ukazování, ale musí být dobře nastavené zahazování pixelů, protože jinak počítač nestíhá obraz zpracovávat.

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/>

Blog se zprávami z aktuálního postupu: <http://zpracovaniobrazu.blogspot.com/>

Poděkování

Tomáši Gavenčakovi za vedení konference na M&M

Karlu Královi a Lukáši Langerovi za vymyšlení poměrové transformace

Lukáši Langerovi za vymyšlení dvoubodové 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
- zelené 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 nevhodných bodů
 - jas zelené < 60%
 - jas červené > 60%

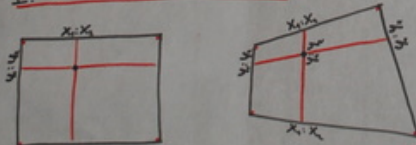
II. Skenování čtverečkem:

- bod osvětlený laserem se zobrazí jako barevný čtvereček o straně $\approx 6px$
- 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

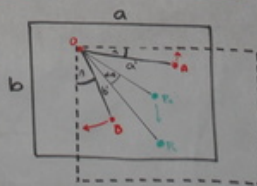
- chceme dataprojektorem zobrazit křížek kolem osvětleného bodu
- protože ale kamera nesnímá obraz z bodu, odkud ho promítá projektor, vzniká zkreslení, které potřebujeme korigovat
- obdélník promítaný dataprojektorem kamera snímá jako obecný čtyřúhelník
- musíme dokázat transformovat \forall bod tohoto čtyřúhelníku na jeho odpovídající bod v obdélníku
- u všech typů kalibrací užíváme kalibrační body v rozích promítaného obrazu

I. Poměrová transformace



- vychází ze zachování poměrů při zkreslení

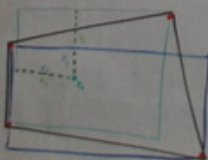
II. Úhlová transformace



$$|OP_1| = |OP_2| = \frac{a}{2} \cdot \frac{b}{2}$$

- vychází z posunutí kalibračních bodů do jejich skutečné polohy a odpovídajícího posunutí bodu P_1

III. Průměrová transformace



- souřadnice bodu P_1 jsou $\left[\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2} \right]$

IV. Grafická transformace pomocí knihovny

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