

Single view metrology

Juhász Réka, Papp László, Pintér Csaba, Soponyai György

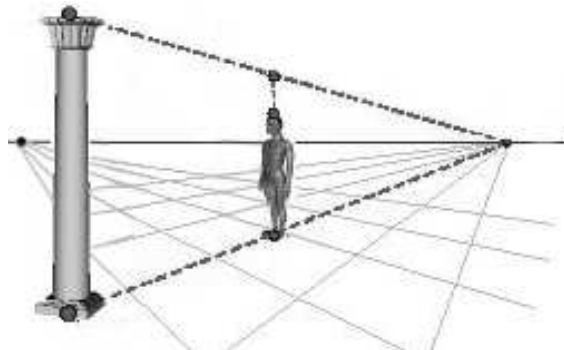
2005. december 22.

A projekt tagjai

név	évfolyam	email cím
Juhász Réka	PTM V.	juhasz.reka.3@stud.u-szeged.hu
Papp László	PTM V.	papp.laszlo.4@stud.u-szeged.hu
Pintér Csaba	PTM V.	pinter.csaba@gmail.com
Soponyai György	PTM V.	soponyai.gyorgy@stud.u-szeged.hu

A megoldandó probléma

A *single view metrology* a számítógépes látás és képfeldolgozás egyik problémája, ahol egy perspektivikus fotó által ábrázolt jelenet térbeli mértékeit szeretnénk meghatározni. Ehhez szükséges megadni a kép egy referenciaobjektumának pontos méretét (pl. az alábbi képen az oszlop magasságát) és ezen ismert adat birtokában meghatározható a kép más objektumainak mérete (az ember magassága).

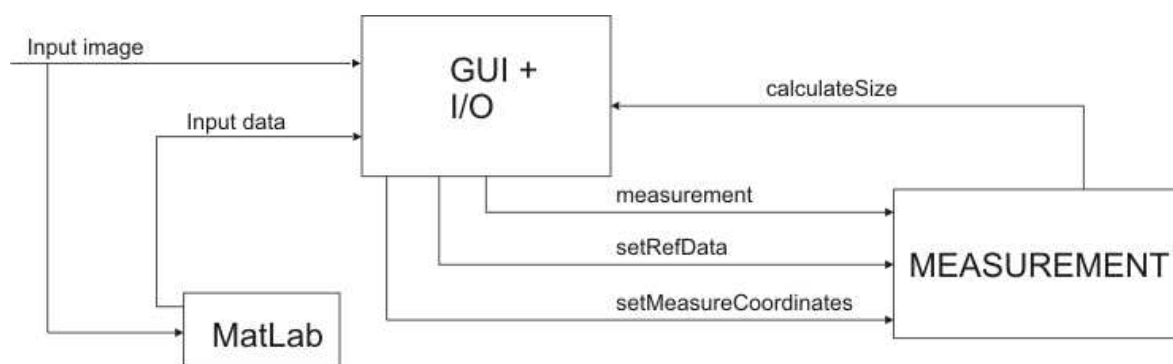


A SVM-rendszer működése

1. A bemeneti képet (*input image*) paraméterként megkapja egyrészt a C++ nyelven megírt grafikus felülettel ellátott kezelőprogram (*GUI+I/O*), másrészt a MatLab.
2. A MatLab egy algoritmus-szkriptet értelmezve meghatározza a kép vanishing point-jait, melyek koordinátáit egy szöveges fájlban (*input data*) elmentve paraméterként megkapja a kezelőprogram.
3. A kezelőprogram irányítja az input/output műveleteket, a felhasználó számára interaktívan biztosítja a képen való objektumkijelöléseket valamint az esetleges input adatok – referenciaobjektum mérete – bevitelét.
4. Az *input data*-t feldolgozza a program, majd egy listában megjeleníti a kép vanishing point-jait, melyből a felhasználó kijelölheti azt a kettőt, melyek együttese megadja a mérés alapjául szolgáló vanishing line-t.

5. Amint a két vanishing point ki van jelölve, a *GUI+I/O* létrehoz egy *MEASUREMENT* objektumot az *svm* konstruktor meghívásával.
6. A képen kijelölünk egy referenciaobjektumot, megadjuk méreteit majd az adatokat átadjuk az *MEASUREMENT* objektumnak a *setRefData* metódus segítségével.
7. Tetszőleges számban kijelöljük a mérendő objektumokat azok tető-, és talppontjaik megadásával, majd átadjuk az adatokat az *MEASUREMENT* objektumnak a *setMeasureCoordinates* metódussal.
8. Az *MEASUREMENT* objektum kiszámolja a mérendő objektum méretét. Az eredményt a *calculateSize* metódus segítségével szolgáltatja a *GUI+I/O* rendszernek, amely megjeleníti az eredményt.¹

Architektúra-terv



MatLab szkript

A programunk által használt MatLab szkript internetes forrásból származik². Feladata egy inputként megadott kép vanishing point-jainak detektálása és a *GUI+I/O* modul számára ezen adatok biztosítása egy fájlban eltárolva. A *VP*-detektálás lépései a következők:

- 1: Az input kép Canny-éldetektálása (*findlines.m*)
- 2: Az él-képre Hough-transzformáció végrehajtása
- 3: A Hough-trafó által generált egyenesek metszéspontjai közül a potenciális vanishing point-ok kigyűjtése (*findvanishingpoints.m*)

A teljes algoritmus az *(általunk implementált)* *vanishing_point.m* szkript elindításával futtatható, amely a következő 4 paraméterrel rendelkezik:

1. A beolvasandó kép neve
2. Az output fájl; a detektálandó vanishing pointok tárolási helye *(a fájl specifikációját lásd lejjebb)*
3. $\begin{cases} \text{true} & \text{színes kép esetén} \\ \text{false} & \text{szürkeárnyaltos kép esetén} \end{cases}$
a paraméter feltüntetése azért szükséges, mert színes (*RGB*) képek esetén el kell végeznie a programnak egy szürkeárnyaltosra konvertálást...

¹Az utolsó két lépés tetszőleges sokszor elvégezhető

²[http://www.dia.uniroma3.it/~iarusso/Seminari%202004 2005/TesinePrededenti/aa0405](http://www.dia.uniroma3.it/~iarusso/Seminari%202004%202005/TesinePrededenti/aa0405)

4. $\begin{cases} \text{true} & \text{ha a kép által ábrázolt jelenet zárt térben (egy szobában) fekszik} \\ \text{false} & \text{„szabadtéri” képek esetében} \end{cases}$

A szkript egy lehetséges paraméterezése a következő:

```
vanishing_point('fenykep1.jpg', 'fenykep1.dat', false, false);
```

Előállhat olyan eset, amikor a vanishing point-ok nem jó pozíciókban helyezkednek el a detektálás pontatlansága miatt. Ekkor a felhasználónak lehetősége adódik arra, hogy a vanishing line-t meghatározó pontokat kézzel, egérgattintással helyezze el.

Az előállított fájl formátumának specifikációja

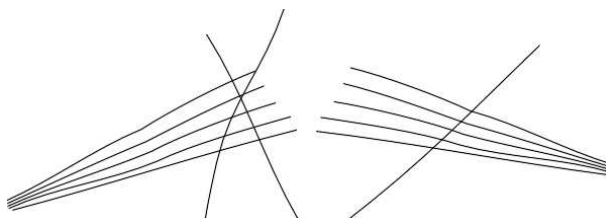
A generált fájl egész számokat tartalmaz legalább 2 sorban. Minden sora két egész számot tartalmaz szóköz-karakterrel elválasztva. Az i -edik sor a kép i -edik vanishing point-jának pixeles koordinátáit tartalmazza. Az x és y koordináták lehetnek bal alsó origó-, vagy mátrix-módszerrel (x balra, y fel / x . sor, y . oszlop) generáltak.

Egy példa adatfájl (A mellékelt `Krakko20.jpg` kép vanishing point-jai):

```
234 456
-1 356
120 765
```

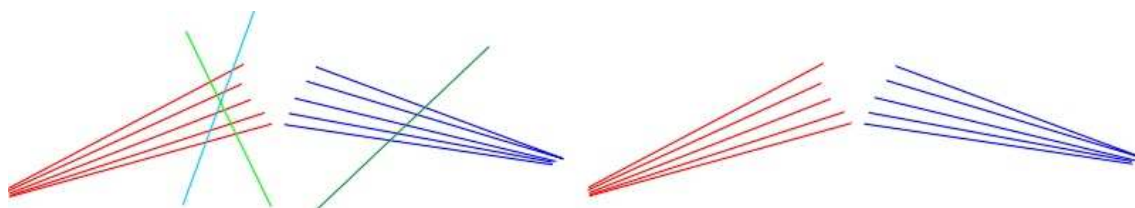
A MatLab-szkript működési elve

A képre először végrehajtunk egy Canny-éldetektálást. A Canny-éldetektor a ma ismert éldetektorok egyik legjobbjika. Mivel az eredménykép 1 pixel vastagságú vonalakat tartalmaz csupán, nem kell különbséget tenni vastag és vékony vonalak között.



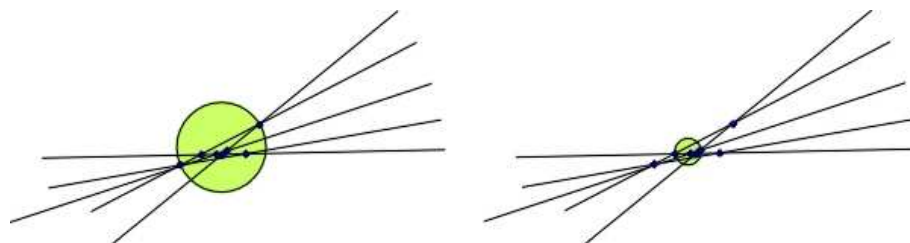
A Canny-éldetektálás eredménye

Az „éldetektált” képre végre kell hajtani egy Hough-transzformációt. A Hough-transzformáció eredményeként előállt egyeneseket halmazokba soroljuk. Ha két adott egyenes között a szög kicsi, akkor egy halmazba kerülnek, különben új halmazt nyitunk. A besorolás után töröljük azokat a halmazokat, amelyek nagyon kevés elemszámúak.



A halmazokba sorolt egyenesek törlés előtt és törlés után

Az élekhez tartozó egyenesek egy perspektivikus képen általában nem egy pontban metszik egymást a raszteres ábrázolásból eredő pontatlanság miatt. Ezért az „egy pontban metszés” fogalmat úgy kell módosítanunk jelen esetben, hogy „egy adott sugarú körben metszés”. E kör sugara annál nagyobb, minél távolabb esik a középpontja a kép középpontjától, mert annál nagyobb a pontatlanság. Ezzel nem kell törődnünk, mert minél távolabb vannak a metszéspontok, annál kevésbé számít a hibás pozíciójuk. Az *isdoor*-paramétert `true`-ra állítva az algoritmus feltételezi, hogy sokkal kevesebb egyenes áll rendelkezésre, így a körök méretei kisebbek. A módszer alkalmazásának hatására a vanishing point-ok azon körök középpontjai lesznek, melyek „*nagyon sok*” egyenes metszéspontjaiként állnak elő.



Képközépponttól távoli és közeli körök mérete

Az algoritmus a három legvalószínűbb vanishing point-ot adja eredményül. Egy *VP* annál valószínűbb, minél több egyenes metszéspontja.

Koordináta-geometriai alapok

A két ponton átmenő egyenes egyenlete $(y - y_1)(x_2 - x_1) = (x - x_1)(y_2 - y_1)$. Mivel egy egyenes megadható az $y = Mx + b$ általános alakban, a két összefüggés alapján meghatározható a következő két tulajdonság:

$$M = \frac{(y_2 - y_1)}{(x_2 - x_1)} \quad b = \frac{y_1(x_2 - x_1) - x_1(y_2 - y_1)}{x_2 - x_1}$$

Függőleges egyenesek esetében a módszer ilyen formában nem alkalmazható a 0-nevező miatt. Ekkor vektoros vagy polárkoordinátás reprezentációt használva a probléma megoldódik. Legyen két, egymással nem párhuzamos egyenes egyenlete a következő:

$$y = M_1x + b_1 \quad y = M_2x + b_2$$

Ezek alapján meghatározható a két egyenes (x_0, y_0) metszéspontja, ahol...

$$y_0 = \frac{M_1b_2 - M_2b_1}{M_1 - M_2} \quad x_0 = \frac{y_0 - b_1}{M_1}$$

Mérés

A fenti koordináta-geometriai tulajdonságokat felhasználva meghatározható a referencia objektum paramétereinek ismeretében a mérendő objektum magassága. Legyen...

- *meretMerendo* a mérendő objektum mérete pixelben,
- *meretMerendoVanish* a mérendő objektum talppontja és az (x_{MV}, y_{MV}) pont távolsága pixelben,
- *resize* az általunk megadott referenciaobjektum mérete.

Ekkor a mérő algoritmus a következő:

- 1: húzzunk egy egyenest a mérendő objektum és a referenciaobjektum talppontjain át

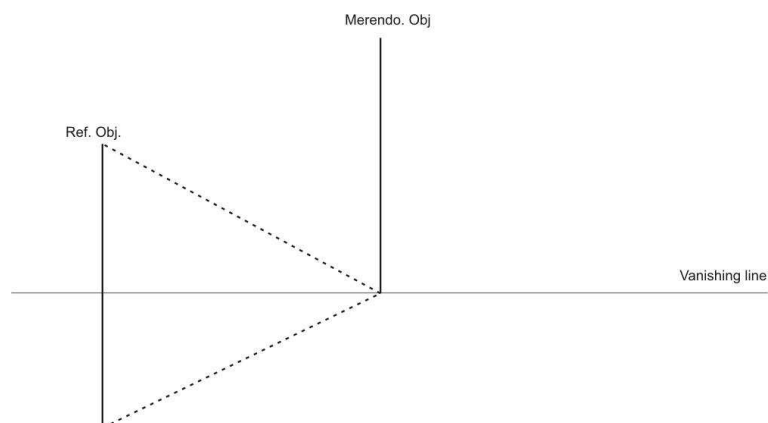
- 2: **if** az egyenes iránytangense megegyezik a vanishing line iránytangensével **then**
- 3: egyszerű arányszámítással meghatározzuk a méretet
- 4: **else**
- 5: meghatározzuk a vanishing line és az egyenes metszéspontját $\rightarrow (x_t, y_t)$ ³
- 6: (x_t, y_t) -n és a referenciaobjektum tetején keresztül húzzunk egy egyenest
- 7: meghatározzuk ezen egyenes és a mérendő objektum egyenesének metszéspontját $\rightarrow (x_{MV}, y_{MV})$
- 8: Arányszámítással meghatározzuk a mérendő objektum méretét: $\frac{\text{refsize} * \text{meretMerendo}}{\text{meretMerendoVanish}}$
- 9: **end if**
- 10: **return** a mérendő objektum metrikus mérete

Abnormális eredmények generálódhatnak „majdnem párhuzamos” egyenesek mérésekor. Vegyük azt a speciális esetet, amikor a mérendő-, és a referenciaobjektum talppontjainak y koordinátái között csupán 1 pixelnyi eltérés van. Ekkor – kellően nagy méretű kép esetén – előfordulhat, hogy a talppontokat összekötő egyenes és a vanishing line metszéspontjának x koordinátája nagyobb (kisebb) lesz, mint az adott C++ fordító által ábrázolható legnagyobb (legkisebb) szám.

Ilyenkor szintén feltesszük, hogy a két objektum egymás mellett áll. Az elfogadott metszéspont értékét `MIN_LONGINT` és `MAX_LONGINT` között határoztuk meg, így biztos minden fordítóval generált kód helyesen fut.

A program hiányosságai

Abban az esetben adódik probléma, ha referenciaobjektum a vanishing line-on „áll”, vagyis talppontja megegyezik (x_t, y_t) -vel⁴. Ekkor a mérendő objektum magassága mindig $+\infty$ lesz a referenciaobjektum méretétől függetlenül:



Továbbfejlesztési lehetőségek

A program több irányba is továbbfejleszthető:

- A program jelen változata a vanishing line-ra merőleges irányt veszi „függőleges”-nek, a számításokban nem veszi figyelembe a függőleges vanishing point elhelyezkedéséből adódó perspektivikus torzítást. Egy későbbi verzióban bevezethető a *függőleges referenciairány* fogalma, ami pontosabb mérést tenne lehetővé.
- MatLab szkript automatikus elindítása a C++ programból egy *wrapper* felület alkalmazásával. Így a felhasználónak csak egyetlen felületet kell kezelnie.
- A kijelölendő objektumok határainak pontosabb megadásához hasznos lehet a *zoomolás*-funkció bevezetése.

³ a „t”-index a „talp”-szóbból származik

⁴ Ugyanígyen helyzet áll elő, ha a mérendő objektum áll a vanishing line-on.

Az *MEASUREMENT* osztály specifikációja

```
class measurement{
private:
    int width;      //kép szélessége
    int height;    //kép magassága

    int vp1_x;     //1. vanishing point x koordinátája
    int vp1_y;     //1. vanishing point y koordinátája

    int vp2_x;     //2. vanishing point x koordinátája
    int vp2_y;     //2. vanishing point y koordinátája

    int reftop_x;  //ref.obj. tetejének x koordinátája
    int reftop_y;  //ref.obj. tetejének y koordinátája

    int reftdown_x; //ref.obj. aljának x koordinátája
    int reftdown_y; //ref.obj. aljának y koordinátája

    float reftsize; //referencia objektum mérete

public:
    measurement (int w, int h, int vp1_x, int vp1_y, int vp2_x, int vp2_y);

    void setRefData (int reftop_x,
                    int reftop_y,
                    int reftdown_x,
                    int reftdown_y,
                    float reftsize);

    void setMeasureCoordinates (int mtop_x, //mérendő tetejének x koord.
                                int mtop_y, //mérendő tetejének y koord.
                                int mdown_x, //mérendő aljának x koord.
                                int mdown_y); //mérendő aljának y koord.

    float calculateSize (); //mérendő objektum méretét meghatározó metódus
};
```

A program fordítása

A program fordításhoz Visual Studio 6.0 fejlesztőkörnyezetre és a WxWindows (vagy WxWidgets) API-ra van szükség. A Visual C++ *Project Settings* menüpontjában a következő paramétereket kell beállítani:

- *C/C++* \Rightarrow *Preprocessor* \Rightarrow *Additional include directories* — Írjuk be a WxWindows könyvtáron belüli `/include` és `/lib/msw` könyvtárak elérési útjait.
- *Link* \Rightarrow *Input* \Rightarrow *Object/Library modules* — A következő bejegyzéseket írjuk ide:
`.../lib/zlib.lib .../lib/regex.lib .../lib/tiff.lib .../lib/png.lib .../lib/jpeg.lib`
`.../lib/wxmsw.lib`
- *Resources* \Rightarrow *Additional resource include directories* — Írjuk be: `.../include`

A fenti beállítások csak *Release* módra érvényesek. *Debug* módnál annyi különbség van, hogy a megfelelő helyeken `msw` helyett `mswd` bejegyzést kell írni, valamint minden *lib*-név végére egy „d”-karakter kerül (pl.: `lib/jpegd.lib`).

A program használati utasítása

A program elindulása után be kell tölteni a képet (*File/Open image*) illetve a vanishing point-okat tartalmazó fájlt (*File/Open VP File*). A kép betöltése után a pontokat nemcsak fájlból tölthetjük be, hanem kézzel is megadhatjuk a kezelőfelület jobb oldalán található *Add Vanishing Points* gombra kattintva. Ebből a funkcióból a *Stop Adding* gomb megnyomására léphetünk ki.

A mérés megkezdéséhez szükségünk van a vanishing line-ra, amit két vanishing point kiválasztásával jelölhetünk ki a programablak jobb oldalán található listából. A képen ekkor megjelenik a két VP zöld kereszt formájában (amennyiben a képen van), valamint az általuk meghatározott vanishing line, ami egy zöld egyenesként ábrázolódik.

Ezután kattintsunk a kijelölendő referenciaobjektum talppontjára, majd az egér gombját lenyomva tartva húzzuk a mutatót az objektum tetőpontjához és engedjük fel a gombot. A megjelenő kék szakasz mutatja a kijelölt objektum képbeli pozícióját. Most a *Ref. object size* mezőbe írjuk be az objektum egy metrikus egységbeli méretét. Kattintsunk a *Define ref. object* nyomógombra, aminek hatására a szakasz piros színűre változik, a végpontjainál kereszttek jelennek meg. Ezzel a referenciaobjektum definiálását befejeztük.

A mérendő objektum kijelölése az előzőekhez hasonló módon történik annyi különbséggel, hogy neki nem kell méretet megadni (*ugyanis épp erre vagyunk kíváncsiak*), és definiálni sem kell RF-ként az erre szolgáló gombbal. Az objektum – kijelölése után – kék szakaszként látszik.

Ezt követően a *Measure* gombra kattintva hajtódik végre a mérés, aminek eredménye a gomb alatti *Measurement* mezőben jelenik meg piros színnel.