# I9 – Teleoperáció és távérzékelés virtuális cellában

*Mérési útmutató az*
*Irányítástechnika és Képfeldolgozás*
*Laboratóriumhoz: 9. mérés*

*Dr. Vajda Ferenc*
*3D érzékelés és Mobilrobotika kutatócsoport*
*Irányítástechnika és Informatika Tanszék*
*2015. október 7.*

Jelen dokumentum a Budapesti Műszaki és Gazdaságtudományi egyetem Irányítástechnika és képfeldolgozás laboratórium 1. c. tárgyához kapcsolódó mérési útmutató. A mérés és az útmutató a laboratórium 9. méréséhez készült a tanszék 3D érzékelés és mobilrobotika kutatócsoportjában.

A mérések és a mérési útmutató kidolgozásában részt vettek a kutatócsoport tagjai:

Vajda Ferenc
Takács Tibor
Srp Ágoston Mihály
Helfenbein Tamás
Tóth András



Verziók:

| Dátum | Verzió | Módosította | Leírás |
|---|---|---|---|
| 10/14/09 | V0.1 | Vajda Ferenc | Első kész változat |
| 11/25/14 | V0.2 | Szemenyei Márton | Második változat |
| 09/02/15 | V0.3 | Szemenyei Márton | Harmadik kész változat |

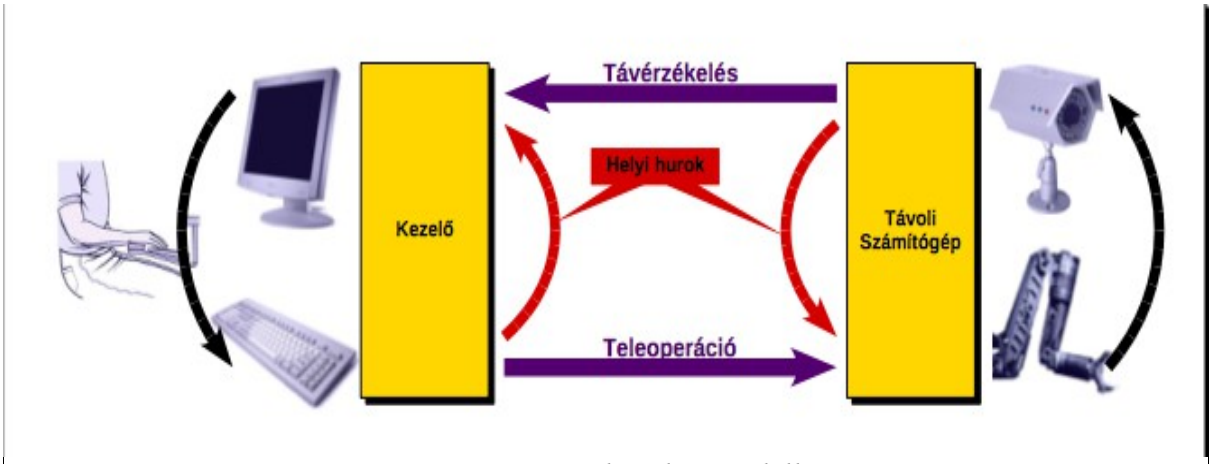
# Tartalomjegyzék

# 1. Teleoperáció és távérzékelés

A valós életben számos olyan problémával találhatjuk szemben magunkat, ahol a feladat elvégzését megnehezíti, hogy a munkavégzés környezete (a munkatér) az ember számára megközelíthetetlen, vagy éppenséggel veszélyes. Ilyen esetekben elterjedt megoldás az iparban a teleoperáció, vagyis távoli, biztonságos környezetből irányított robotok használata az adott feladat megoldásának érdekében. Az ilyen esetekben alkalmazott robotok gyakran rendelkeznek a környezetük feltérképezését, vagy egyéb információgyűjtést szolgáló szenzorokkal is, melyek adatai a robotot kezelő irányító döntéseit segítik. Ezt a folyamatot nevezzük távérzékelésnek.

Gyakori probléma azonban a távérzékelés és a teleoperáció használata során, hogy a munkatér és az irányítóközpont közti nagy fizikai távolság, vagy egyéb technológiai korlátok az ember számára is érzékelhető és a kezelőket zavaró mértékű késleltetést okoznak. További problémaforrás lehet, ha a robot és a vezérlőközpont közti kommunikációs megoldás limitációi korlátozzák az átvihető adatmennyiséget, így nincs lehetőségünk az összes szenzoradat továbbítására az irányító felé.

A fenti problémák miatt indokolt lehet a rendszer munkatérbeli, illetve a kezelőtérben található részegységében is egy visszacsatolás segítségével helyi irányítóhurkokat létrehozni. A helyi hurkok segítségével mindkét oldalon lehetőség nyílik a másik résztvevő által szolgáltatott jelek becslésére, szimulációjára késleltetés nélkül, az esetleges hibák korrigálására pedig a valódi adatok beérkezésekor lehetőség nyílik. A távérzékelés és teleoperáció modelljét a helyi visszacsatolásokkal kiegészítve kapjuk a Sheridan-modellt (1. ábra).




***1. ábra:*** *A Sheridan-modell*

A távoli számítógép oldalán lévő helyi hurok esetén az irányítás, döntéshozás, valamint a jelfeldolgozás megvalósítása szükséges, így a robot képes váratlan helyzetekben az irányító beavatkozása (és így késlekedés) nélkül reagálni. A távoli hurokban lehetőség nyílik az irányító jövőbeli döntéseinek becslésére is, a használt módszerek megbízhatósága azonban sok esetben nem megfelelő.

A kezelői oldalon található hurok feladata a robotnak, illetve a környezetének szimulációja, így azonnali visszajelzést képes szolgáltatni a kezelőnek a robot feltételezett reakciójáról a kiadott vezérlésre. Ezt a feladatot gyakorta végezzük el virtuális valóság környezetek segítségével, melyben mind a robot, mind a környezet modelljét elhelyezzük. A kezelői oldalon található vezérlési hurok fontos feladata, hogy a szimuláció eredményét a beérkező adatok alapján pontosítsa, nagy mértékben javítva a kezelhetőséget.

# 2. Objektumfelismerés, pozíciómeghatározás

A kamerarendszerek egyik legfontosabb felhasználási célja a környezetben található objektumok szétválasztása és a lényeges elemek kiemelése, más szóval szegmentálás. Tekintettel a fontosságára, illetve a feladat meglehetősen nem egzakt jellegére, számos jó és kevésbé jó algoritmust alkalmazhatunk. Legtöbbjük csak különleges körülmények között használható, vagy olyan mértékű futásidőigénye, hogy a legtöbb alkalmazásban szóba se kerülhet. Emiatt az alkalmazások nagy részénél valamilyen módon egyszerűsítenünk kell a környezetet, és olyan módon kell átalakítanunk, hogy az algoritmus számára könnyedén feldolgozható legyen. Erre megoldás, ha a megkeresendő objektum színét megváltoztatjuk olyan módon, hogy a környezettől elüssön.

A szegmentált objektumok megtalálása után annak releváns tulajdonságait is meg kell határoznunk. Szinte minden esetben szükséges a pozíció és/vagy orientáció, de gyakran van szükség méretre stb. A lényeges tulajdonságok kiemelése általában egyszerűbb feladat, mint a szegmentálás, de sokszor ezek meghatározása is szinte lehetetlennek tűnik.

Az alábbiakban ismertetünk néhány fontosabb algoritmust, amely segítségével a mérésben szükséges képfeldolgozási feladatok elvégezhetők. Ez természetesen nem jelenti, hogy a hallgató nem használhat szofisztikáltabb megoldást.

## 2.1 Objektumtulajdonságok meghatározása

A szegmentálás során megtalált objektumok azonosításához meg kell határoznunk az objektumok paramétereit. Ezek közé tartoznak az objektum mérete, elhelyezkedése, alakja, mintázata, színe stb. Az alábbiakban a mérés szempontjából fontosabb tulajdonságokra összpontosítunk

### 2.1.1 Objektumtulajdonságok nyomatékok alapján

Az objektumok területének, pozíciójának és orientációjának meghatározásához

felhasználhatjuk a különféle geometriai nyomatékokat. Ehhez szükséges, hogy az adott képen csak az objektum legyen látható. A megoldás működik bináris, és szürkeárnyalatos képeken is.

A hallgatók által is használt OpenCV rendszerben az egyes nyomatékok elnevezései $m_{ij}$ formában jelennek meg, így ezek is megtalálhatók az alábbiakban. Tekintettel arra, hogy a harmad, vagy magasabb rendű nyomatékok használata igen ritka, ezekkel a továbbiakban nem foglalkozunk.

### Nulladrendű nyomaték

A nulladrendű nyomaték a megtalált objektum területét adja meg.

$$A = \sum_x \sum_y I(x,y) = m_{00} \tag{1}$$

### Elsőrendű nyomaték

Az elsőrendű nyomaték az objektum statikai nyomatékát adja meg. Az elsőrendű és a nulladrendű nyomaték hányadosából megkaphatjuk az objektum súlypontjának x, illetve y koordinátáját.

$$S_x = \sum_x \sum_y y \cdot I(x,y) = m_{01} \tag{2}$$

$$S_y = \sum_x \sum_y x \cdot I(x,y) = m_{10} \tag{3}$$

Ebből

$$x_c = \frac{S_y}{A} = \frac{\sum_x \sum_y x \cdot I(x,y)}{\sum_x \sum_y I(x,y)} \tag{4}$$

$$y_c = \frac{S_x}{A} = \frac{\sum_x \sum_y y \cdot I(x,y)}{\sum_x \sum_y I(x,y)} \tag{5}$$

### Másodrendű nyomaték

A másodrendű nyomatékok az objektum x és y tengely körüli „tehetetlenségi”

nyomatékát adja meg. Orientációként többnyire a legkisebb másodrendű nyomatékhoz tartozó tengely irányát szoktuk választani.

$$J_x = \sum_x \sum_y y^2 \cdot I(x,y) = m_{02} \tag{6}$$

$$J_y = \sum_x \sum_y x^2 \cdot I(x,y) = m_{20} \tag{7}$$

Illetve a Centrifugális másodrendű nyomaték:

$$J_{xy} = \sum_x \sum_y xy \cdot I(x,y) = m_{11} \tag{8}$$

A nyomatékok alapján a súlyponton átmenő legkisebb nyomatékú tengely iránya (a részletes magyarázatra itt nem térünk ki):

$$a = J_y - \frac{S_y^2}{A} \tag{9}$$

$$b = 2 \cdot \left( J_{xy} - \frac{S_x S_y}{A} \right) \tag{10}$$

$$c = J_x - \frac{S_x^2}{A} \tag{11}$$

Amelyből:

$$\tan 2\theta = \frac{b}{a-c} \tag{12}$$

$$\sin 2\theta = \pm \frac{b}{\sqrt{b^2 + (a-c)^2}} \tag{13}$$

$$\cos 2\theta = \pm \frac{a-c}{\sqrt{b^2 + (a-c)^2}} \tag{14}$$

Azt a szöget kell megkeresnünk, amelyre mindhárom egyenlet teljesül. Tekintettel arra, hogy a tengely elfordulási szögét keressük, ennek értéke 0-180° közé eshet.

Érdemes még megemlíteni a középpontra vett másodrendű nyomatékot

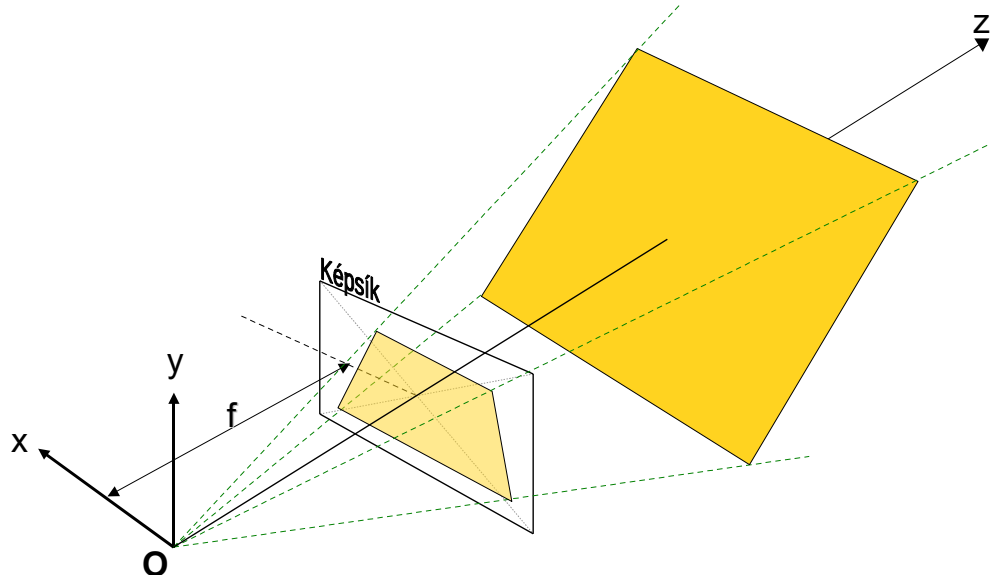
$$J_0 = \sum_x \sum_y (x^2 + y^2) \cdot I(x,y) = J_x + J_y \tag{15}$$

amely alakinformációkat hordoz. Ez is abban az esetben a leghasznosabb, ha a középpont nem a kép (0,0) pozíciója, hanem az objektum súlypontja:

$$J_{0,c} = J_0 - \frac{S_x^2 + S_y^2}{A} \tag{16}$$

## 2.2 Kétdimenziós pozíciómeghatározás a perspektív torzítású képen

Amennyiben egy kamera nem pontosan felülről látja a munkateret, a munkatér a képen torzítva jelenik meg. Igaz azonban az, hogy amennyiben a munkatérben vizsgált összes objektum egy síkban helyezkedik el, a képernyőn található pontok egyértelműen összefüggenek a valós pontokkal. Az összefüggést a perspektív transzformáció adja meg.




**2. ábra** *Perspektív transzformáció két dimenzióban*

A transzformáció homogén koordináták segítségével az alábbi módon fogalmazható meg:

$$\begin{bmatrix} x_w \\ y_w \\ w \end{bmatrix} = \mathbf{T}_{\text{persp}} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{17}$$

Ahol $u$ és $v$ a képkoordináták, a 3×3 $\mathbf{T}_{\text{persp}}$ pedig a kamera és a viszonyítási koordinátarendszer közötti forgatást és eltolást, illetve az egy ponton keresztüli síkra történő vetítés transzformációs együtthatóit tartalmazza. Ebből a világkoordináták az

alábbi módon származtathatók.

$$x=\frac{x_w}{w}, \text{ ill. } y=\frac{y_w}{w} \tag{18}$$

Az ellenkező irányú transzformációt is elvégezhetjük a perspektív operátorral:

$$\begin{bmatrix} u_s \\ v_s \\ s \end{bmatrix} = \mathbf{T}_{\mathbf{persp}}^{\mathbf{W\rightarrow C}} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{19}$$

illetve

$$u=\frac{u_s}{s}, \text{ és } v=\frac{v_s}{s} \tag{20}$$

Mivel nekünk a kamerakép áll rendelkezésünkre, a továbbiakban a (17)-(18) egyenleteket használjuk. A perspektív transzformációs mátrix meghatározásához fel kell vennünk pontokat, amelyeknek ismerjük a pontos pozícióját a valós térben és a kameraképen is. Tekintettel arra, hogy $w$ értéke tetszőleges lehet (nyilvánvalóan leszámítva a 0 értéket) a mátrix determinánsa is tetszőlegesen megválasztható, vagyis a kilenc érték nem független egymástól, egyet tetszőlegesen megválaszthatunk. Az egyszerűség kedvéért válasszuk $t_{33}=1$ értéket. A többi paraméter azonban meghatározható. Mivel nyolc paraméter értékére vagyunk kíváncsiak, nyolc független összerendelésre van szükségünk. Mivel minden pont két független értéket tartalmaz (x, y) → (u, v), négy ismert pontra van szükségünk. Minden pontra igaz (17) alapján, hogy

$$\begin{bmatrix} x_{w,*} \\ y_{w,*} \\ w \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & 1 \end{bmatrix} \cdot \begin{bmatrix} u_* \\ v_* \\ 1 \end{bmatrix} \tag{21}$$

Vagyis

$$x_*=\frac{t_{11}u_*+t_{12}v_*+t_{13}}{t_{31}u_*+t_{32}v_*+1}, \; ill. \; y_*=\frac{t_{21}u_*+t_{22}v_*+t_{23}}{t_{31}u_*+t_{32}v_*+1} \tag{22}$$

Amelyből

$$\begin{aligned} u_*\cdot t_{11}+v_*\cdot t_{12}+1\cdot t_{13} \qquad & -u_*x_*\cdot t_{31}-v_*x_*\cdot t_{32}=x_* \\ u_*\cdot t_{21}+v_*\cdot t_{22}+1\cdot t_{23} & -u_*y_*\cdot t_{31}-v_*y_*\cdot t_{32}=y_* \end{aligned} \tag{23}$$

Az összes pontra lineáris formában felírva

$$\mathbf{A}\cdot\mathbf{t}=\begin{bmatrix} u_0 & v_0 & 1 & & & & -u_0x_0 & -v_0x_0 \\ & & & u_0 & v_0 & 1 & -u_0y_0 & -v_0y_0 \\ u_1 & v_1 & 1 & & & & -u_1x_1 & -v_1x_1 \\ & & & u_1 & v_1 & 1 & -u_1y_1 & -v_1y_1 \\ u_2 & v_2 & 1 & & & & -u_2x_2 & -v_2x_2 \\ & & & u_2 & v_2 & 1 & -u_2y_2 & -v_2y_2 \\ u_3 & v_3 & 1 & & & & -u_3x_3 & -v_3x_3 \\ & & & u_3 & v_3 & 1 & -u_3y_3 & -v_3y_3 \end{bmatrix}\cdot\begin{bmatrix} t_{11} \\ t_{12} \\ t_{13} \\ t_{21} \\ t_{22} \\ t_{23} \\ t_{31} \\ t_{32} \end{bmatrix}=\begin{bmatrix} x_0 \\ y_0 \\ x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \end{bmatrix}=\mathbf{b} \tag{24}$$

Így megkaphatjuk a transzformációs paramétereket:

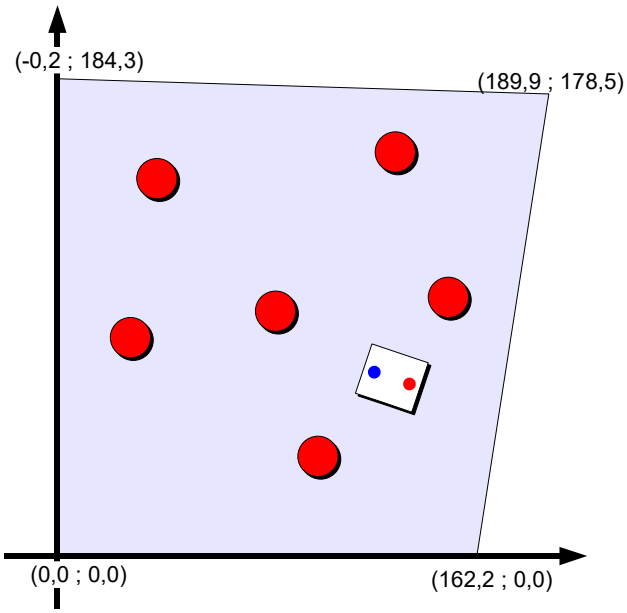
$$\boxed{\mathbf{t}=\mathbf{A}^{-1}\cdot\mathbf{b}} \tag{25}$$

# 3. Teleoperáció bemerüléses virtuális valóság segítségével

A mérés során a hallgatók megismerkednek az objektumfelismerés és a pozíciómeghatározás alapjaival, illetve betekintést nyernek a virtuális valóság rendszerekben való alkalmazásaiba.

## 3.1 Célkitűzések

A mérés során a hallgatók egy sík munkaterületen dolgoznak, melyben egy távirányítású jármű, valamint egységes kinézetű akadályok helyezkednek el (3. ábra). Ezt a munkaterületet egy kamera figyelmi meg felülről/oldalról, így a vizuális távérzékelés használatához először a perspektív transzformáció meghatározása szükséges.



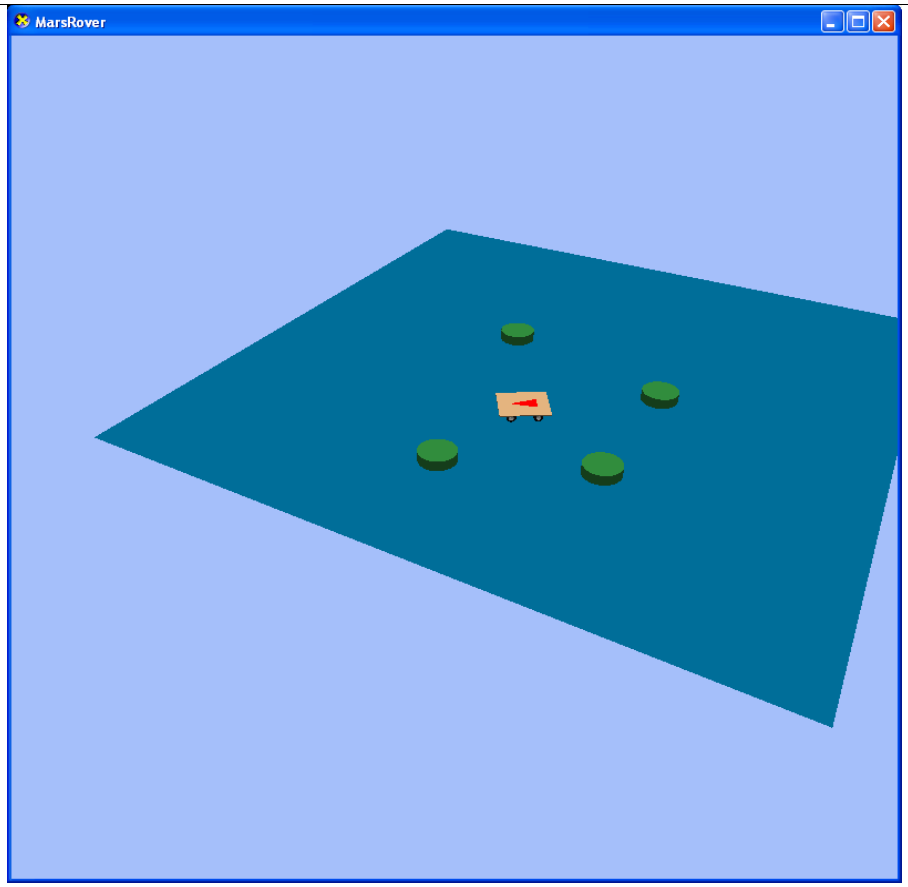

***13. ábra*** *Munkaterület*

Ennek a transzformációnak a meghatározásához a hallgatók egy előre elkészített programot alkalmaznak, mely a munkatér négy kijelölt sarkába elhelyezett akadályok

segítségével végzi el a számításokat. Ehhez a hallgatók két felvételt készítenek a munkatérről: Egy referenciaképet, melyen akadályok nem szerepelnek, valamint egy kalibrációs képet, melyen a négy akadály a munkatér négy kijelölt pozíciójában helyezkedik el. A kalibráció elvégzése után a hallgatók az akadályokat a munkatéren belül tetszőleges pozícióba helyezhetik, majd a program segítségével ezek pozícióit meghatározzák. Ezt követően az akadályokat statikusnak tekintjük.

A mérés következő részében a hallgatók egy saját algoritmusot fejlesztenek, amely a kamera képén a jármű pozícióját és orientációját képes meghatározni a munkatér koordiántarendszerében. Definíció szerint a jármű középpontja a rajta található kék, illetve piros színű köröket összekötő szakasz felezőpontja, az orientációja pedig a prios körből a kékbe mutató vektor x tengellyel bezárt szöge. Az algoritmust az OpenCV nyílt forráskódú képfeldolgozó függvénykönyvtár segítségével valósítják meg. Ebben a feladatban segítséget nyújtanak a 4. fejezetben található minta kódrészletek, amelyek az OpenCV alapvető funkcióit demonstrálják.

A hallgatók által előállított adatok (a jármű, illetve az akadályok pozíciója) egy virtuális valóság alkalmazás (4. ábra) bemeneti adataként szolgálnak, mely lehetővé teszi, hogy a hallgatók a távirányítású robotot mesterséges környezetben vezérelhessék egy távoli számítógépről. A környezet használata során a hallgatók megismerkedhetnek a virtuális környezetek használatának előnyeivel, illetve hátrányaival.




***4. ábra:*** *A virtuális környezet*

## 3.2 A mérés során alkalmazott eszközök

| Eszköz | Típus | Paraméterek | |
|---|---|---|---|
| Kamera | Canon VC-C1 MK II | Felbontás:<br>Típus: | 640x480<br>3 csatornás, RGB |
| Akadályok | Saját | Alapterület:<br>Átmérő: | Kör<br>15,6 cm |
| Robot | Saját | Alapterület:<br>Markerek: | 23×20 cm<br>Kék, piros kör, Ø5cm |
| Munkatér | Saját | Terület: | ~190×185 cm (l. 1. ábra)<br>kamera által látott terület |

## 3.3 Feladatok

a) A méréshez tartozó kamera segítségével készítse el a kalibrációhoz szükséges képeket a munkatérről.

b) A mérésvezető által szolgáltatott program segítségével határozza meg az objektumok középpontjai által a kamera és a sík közötti perspektív transzformáció paramétereit.

c) Írjon rutint, amely a kamera képén megtalálja a robotot, meghatározza annak valós pozícióját és orientációját. Használja az a, pontban előállított paramétereket a Valós értékek meghatározásához.

d) Írjon automata vezérlő rutint, mely megállítja a robotot, ha az akadálynak ütközne.

e) Vizsgálja meg az algoritmus gyorsítására szolgáló lehetséges eljárásokat.

f) Tesztelje az eljárást a mérésvezető által biztosított rendszerrel.

# 4. OpenCV referencia

Az alábbiakban egy nagyon rövid összefoglalást adunk a legfontosabb OpenCV típusokról és függvényekről. Részletesebb információ az OpenCV dokumentációjában található.

## 4.1 Példa-programrészletek

### 4.1.1 Kép létrehozása, manipulálása

```
// szürkeárnyalatos, egycsatornás kép ( a kép alapvetően cv::Mat )
cv::Mat image( cv::Size( width, height ), CV_8UC1 );

// színes, háromcsatornás kép
cv::Mat image( cv::Size( width, height ), CV_8UC3 );

// másik képpel megegyező méretű és típusú kép
cv::Mat image( other.size(), other.type() );

// betöltés fájlból
cv::Mat image = cv::imread( "filename.ext" );

// skálázás fix métetre
cv::resize( original, scaled, cv::Size( newWidth, newheight ) );

//skálázás arányosan
cv::resize( original, scaled, cv::Size( 0, 0 ), xRatio, yRatio );

// képrészlet kivágása
cv::Mat roi = image( cv::Rect( x, y, width, height ) );
```

### 4.1.2 Megjelenítése

```
// kép megjelenítése
cv::imshow( "Ablak neve és egyben fejléce", image );

// várás billentyűzetre
int key = cv::waitKey( 0 ); // várás ideje ms-ban. Ha 0, akkor végtelen
// waitKey LEGALÁBB EGYSZER KELL!!!
```

### 4.1.3 Bináris kép előállítása

```
// szürkeárnyalatos kép előállítása színes képből
cv::Mat gray, binary;
```

```
cv::cvtColor( image, gray, CV_BGR2GRAY );
// BGR és nem RGB formátum az alapértelmezett!!!

cv::threshold( gray, bin, thresholdValue, 255, cv::THRESH_BINARY );
// 255 a küszörbértéket meghaladó pixelek új értéke
```

### 4.1.4 Színtérkonverzió, szétválasztás komponensekre

```
// váltás színterek között
cv::Mat hsvImage;
cv::cvtColor( rgbImage, hsvImage, CV_BGR2HSV ); // más konverziók is lehetnek
// színterek: RGB, BGR, RGBA, BGR555, BGR565, GRAY, YCrCb, HSV, HLS, Luv, Lab
//            XYZ, BayerBG, BayerGB, BayerRG, BayerGR

// komponensek szétválasztása
std::vector< cv::Mat > hsv( 3 );
cv::split( hsvImage, hsv );
```

### 4.1.5 Műveletek objektumokkal

```
//objektumok keresése
cv::FindContours( binImg, contours, CV_RETR_EXTERNAL, CV_CHAIN_APPROX_SIMPLE );

// momentum kiszámolása
for( int i( 0 ); i < contours.size(); i++ ) {
     cv::Moments moments = cv::moments( cv::Mat( contours[ i ] );
     float zerothOrderMoment = moments.m00;
}
```

### 4.1.6 Műveletek képrészletekkel

```
// kép feltöltése 0-val
cv::Mat maskImage( size, type, cv::Scalar( 0, 0, 0 ) );

// kontúrok segítségével maszk kép készítése
for( int i( 0 ); i < contours.size(); i++ ) {
     cv::drawControus( maskImage, contours, i, cv::Scalar( 255, 255, 255 ),
     thickness );
     // thickness a vonal vastagsága, ha -1, akkor kitölti a kontúr belsejét
}

// kép maszkolása a maskImage segítségével
cv::Mat roiImage;
originalImage.copyTo( roiImage, maskImage );
```

### 4.1.7 Perspektív transzformáció

```
// kiinduló pontok megadása
std::vector< cv::Point2f > imagePointVector;
imagePoints.push_back( p1 );
imagePoints.push_back( p2 );

// képek előállítása
cv::Mat iPoints( imagePointVector ), wPoints;

// perspektív transzformáció
cv::perspectiveTransform( iPoints, wPoints, projMatrix );

// pontok előállítása
cv::Point2f P1( wPoints.at< float >( 0, 0 ), wPoints.at< float >( 0, 1 ) );
cv::Point2f P2( wPoints.at< float >( 1, 0 ), wPoints.at< float >( 1, 1 ) );
```

# 5. Ellenőrző kérdések

1. Rajzolja le a távérzékelés és a teleoperáció Sheridan-féle modelljét. Mi a helyi hurkok feladata?
2. Mit jelent egy objektum nullad- és elsőrendű momentuma, illetve milyen összefüggésben áll az objektum középpontjával?
3. Mit ad meg a perspektív transzformáció? Milyen módon határozható meg?
4. Foglalja össze 2-3 mondatban a mérés célját és feladatait!
5. Írja le egy színes kép bináris képpé történő konverziójának lépéseit! (csak műveletek, kód nem kell)
6. Milyen műveleteket végezne el, ha egy bináris képen lévő objektumok méretét szeretné meghatározni? (kód nem kell)
7. Milyen lehetőséget biztosít az OpenCV a perspektív transzformáció elvégzésére?