Gesztusvezérlés

Okosváros laboratórium 6. mérése

Bevezetés

A mérésben egy webkamera gesztusokkal történő vezérléséről van szó. A mérés során rendelkezésre áll egy telepített IP-n keresztül elérhető és vezérelhető webkamera, és egy Leap Motion eszköz, amely alkalmas az ujjak, kezek, karok követésére és ezekkel megvalósított gesztusok azonosítására. A mérés során meg kell ismerni a webkamera vezérlését és a Leap Motion eszközt és annak programozó interfészét. Ezek segítségével és az Ember-gép interfész tárgy keretében tanult tervezési elveket alkalmazva meg kell tervezni egy gesztus alapú felhasználói interfészt. A megtervezett interfészt egy C++ nyelvű példaprogram kibővítésével kell megvalósítani.

Leap motion megismerése [1][2][3]

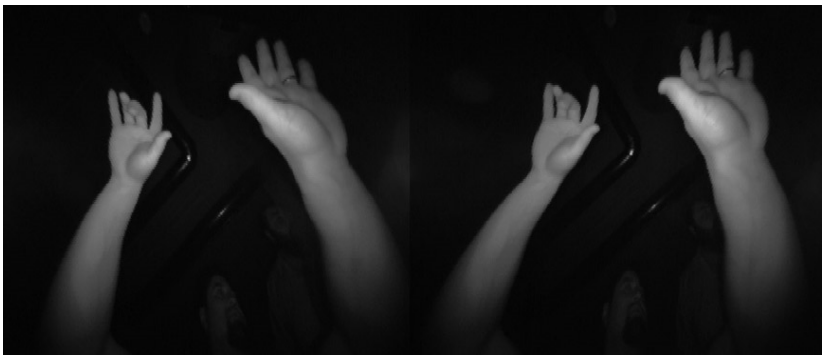
A Leap Motion rendszer felismeri és követi a kezeket, ujjakat és ujj-szerű eszközöket. Az eszköz közelről nagy pontossággal felismeri követi a kezet, folyamatosan elérhetők a pozícióadatok, a felismert gesztusok és mozgások.

Leap Motion elhelyezkedése a kezekhez képest.

Az eszköz optikai alapon működik, 850 nm infravörös tartományban. Ez kívül esik a látható fény spektrumán. Az eszköz kb. 150 fokos tartományban lát. A felismerés az eszköz felett kb. 25 mm-től 600mm-ig működik. Az eszköz akkor működik megfelelően, ha kontrasztos képet kap, ezért az eszköz felületének tisztának kell lennie. Ha ujjlenyomatos, koszos a felismerés hatékonysága csökken. A felismerés a kép és az emberi kézről alkotott belső modell alapján működik.

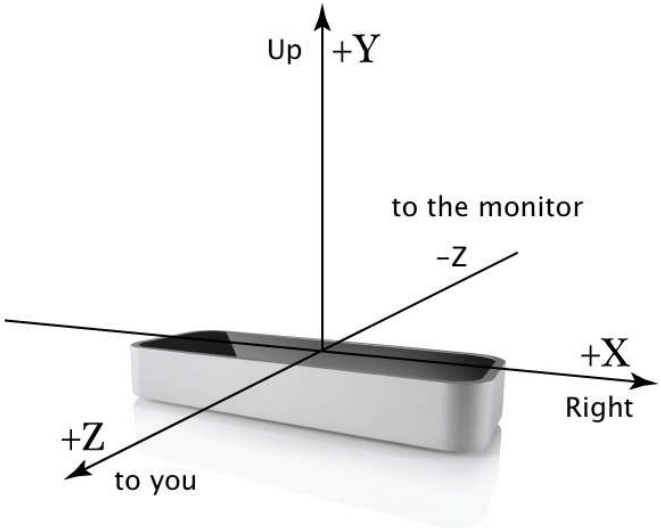
Leap Motion robbantott ábrája.

Az eszközben két infravörös kamera látható. Ezek szürkeárnyalatos sztereó képet készítenek. A képeket többnyire csak az infravörös LED-ek által megvilágított kezek láthatóak, bár más fényforrás is bocsájt ki infravörös fényt.



Két kamera szürkeárnyalatos képe

Koordináta rendszer



Leap Motion által használt koordináta rendszer.

A Leap Motion rendszer egy derékszögű koordináta rendszert használ, amelynek az origója az eszköz felszínének közepén helyezkedik el. Az X tengely jobbra, az Y felfele, a Z tengely pedig a felhasználó felé mutat.

A Leap Motion által mért fizikai mennyiségeket a következő mértékegységekben fejezik ki:

Távolság: mm - milliméter

Idő: µs - mikroszekundum (ha más, akkor külön jelezve van)

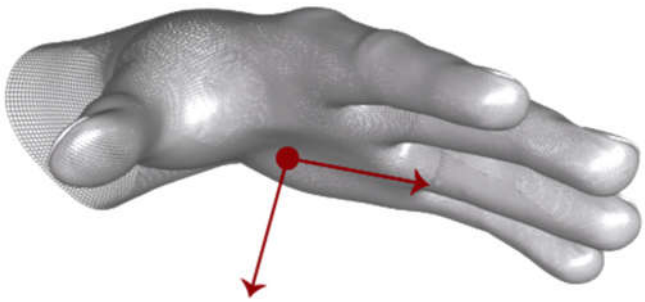
Sebesség: mm/s – milliméter per másodperc

Szög: radián

A mozgás követésének adatait keretenként (frame) adja vissza a rendszer. Minden keretnél a kezek, ujjak és eszközök adatai adja vissza.

Kezek

A kéz modellnél rendelkezésre áll, hogy melyik karhoz csatlakozik és milyen ujjak tartoznak hozzá, illetve a kéz pozíciója és egyéb felismert tulajdonságai



A tenyérből kiinduló normál vektor elhelyezkedése

A belső modell alapján akkor is kapunk különböző pozíció adatokat, ha a kéz csak részlegesen látszik.

Karok

A karok pozíciójáról is rendelkezésre állnak a követési adatok. Amennyiben a könyök nem látható a modell és a korábbi követési adatok alapján megbecsüli a karok pozíció adatait.

Ujjak

A rendszer minden ujjról közöl adatokat. Amennyiben nem látható minden ujj, akkor az anatómiai modell és az aktuális megfigyelések alapján megbecsüli a nem látható ujjak pozícióit.

Az ujjak irányvektorai

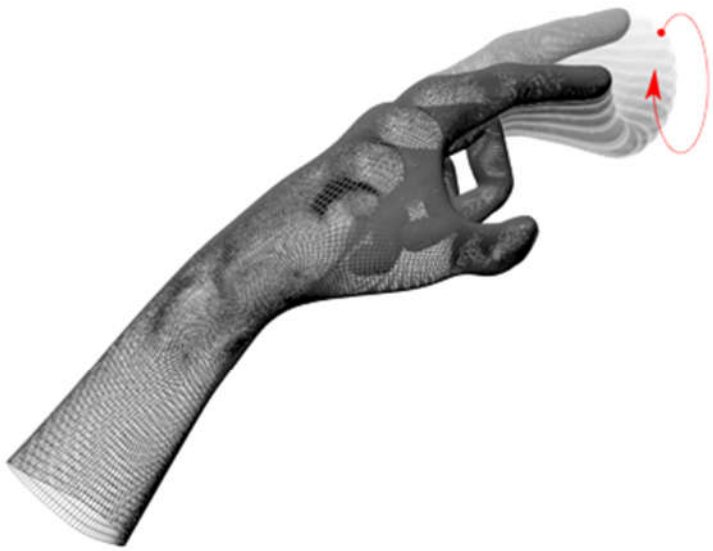
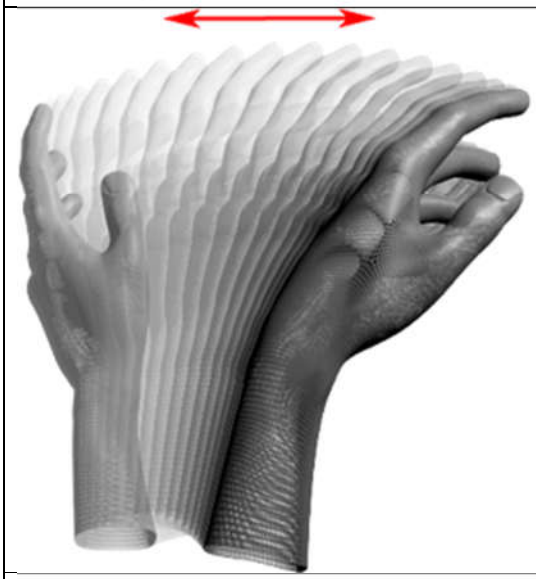
Eszközök

Eszköz lehet például egy ceruza. Az eszközre jellemző, hogy hosszabb, vékonyabb és egyenesebb, mint egy ujj. Csak hengeres eszközöket azonosít a rendszer.

Kéz és egy eszköz.

Gesztusok

A Leap Motion rendszer megadott mozgásmintákat azonosít gesztusként. A gesztusok az ujjak vagy az eszköz megfigyelésén alapulnak. A felismert gesztusok a keretekhez kapcsolódó adatok között jelennek meg. A következő gesztusokat ismeri fel a rendszer alapértelmezetten:

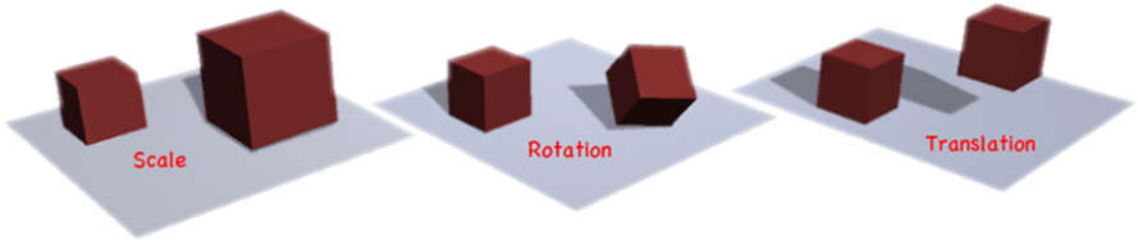
	Circle Egy ujjak körözés
	Swipe Egy hosszú egyenes mozgás a kézzel és az ujjakkal
	Key Tap Egy érintő mozgás, mintha lenyomnák egy gombot a billentyűzeten

	Screen Tap Egy érintő mozgás, mintha megnyomnánk egy függőleges érintő képernyőt

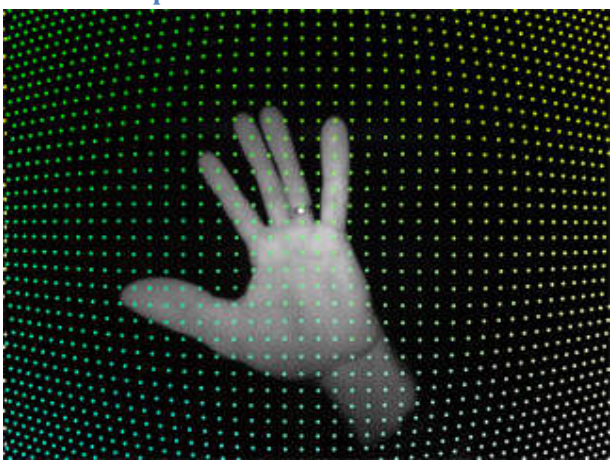
A különböző gesztusok felismerését az API-ban külön engedélyezni kell. A gesztusok felismerésének pontossága függ a gép leterheltségétől, pl. egy debug program kevésbé jól ismeri fel ezeket.

Mozgások

A mozgások a felhasználó kezének adott időegység alatti változásából számítja ki a rendszer. A mozgás lehet 3 fájta lehet:



Érzékelő képei:



A Leap Motion kameráinak nyers adatai is elérhetőek. A kép tartalmazza a mért világosság adatokat is és a kalibrációs adatok, amellyel a lencse torzítás korrigálható.

Leap motion programozása

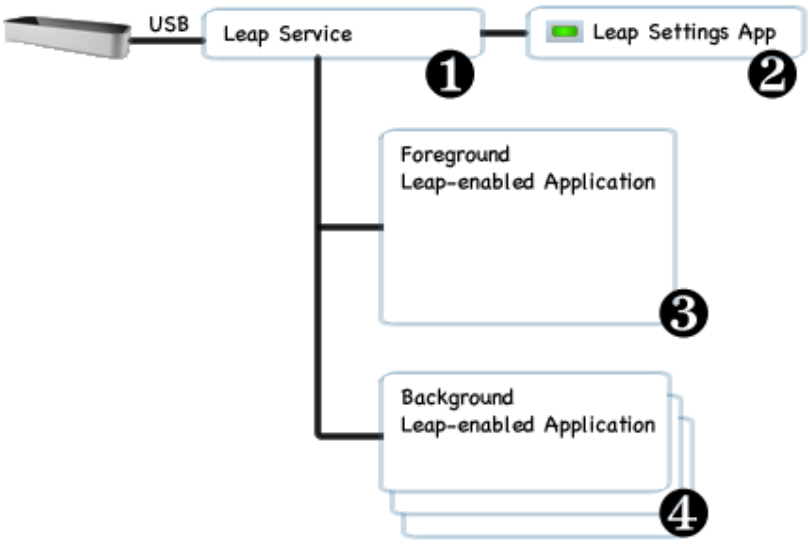
A leap motion SDK számos programozási nyelven és platformon elérhető. A támogatott programozási nyelvek, környzetek:

A felsorolt nyelvek közül a C++ programozási nyelvet fogjuk használni

Leap Motion API:

Natív interfész:

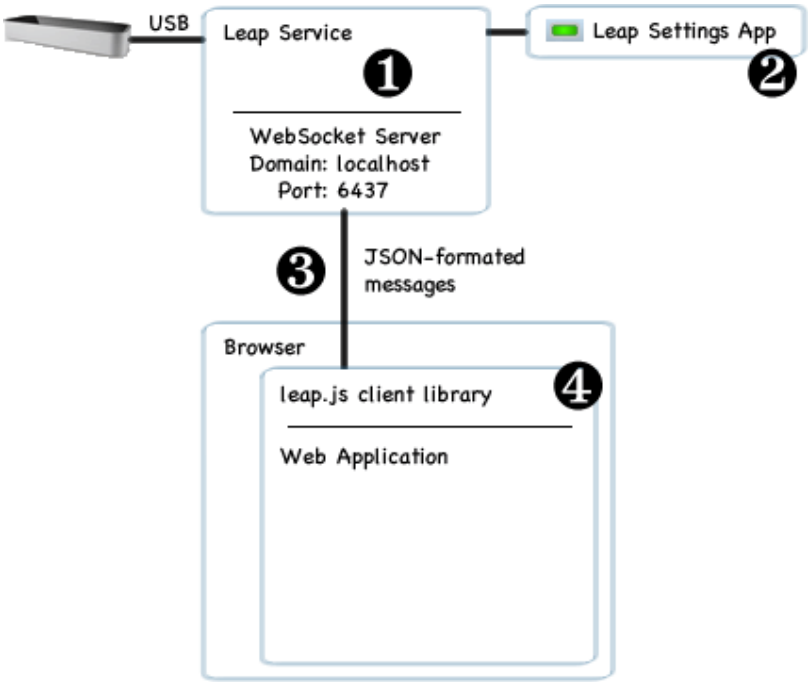
A program dinamikus könyvtáron keresztül kapcsolódik a Leap Service-hez:



1. A Leap Motion szolgáltatás USB-n keresztül kapja az adatokat a kontrollertől. Feldolgozza azokat és küldi tovább a futó alkalmazásoknak.
2. A futó programoktól függetlenül lehet a szolgáltatás paraméterei állítani, külön futó alkalmazással, amely a szolgáltatáshoz kapcsolódik.
3. Az előtérben futó program folyamatosan kapja a mozgási adatokat. Az előtérben futó alkalmazás tud a szolgáltatáshoz kapcsolódni.
4. Amikor a futó alkalmazás elveszti a fókuszt és az operációs rendszer a háttérbe küldi, a szolgáltatás leállítja az alkalmazásnak történő adatküldést. Ha háttérbe került az alkalmazás, kérhet engedélyt az adatok fogadására.

Websocket interfész:

A Leap Motion szolgáltatás a 6437-es porton egy websocket interfészt biztosít:



1. A Leap Motion szolgáltatás websocket szervere elérhető: http://127.0.0.1:6437.
2. A Leap Motion control panel-on engedélyezető vagy letiltható a WebSocket szerver.
3. A szerver JSON formában küldi az adatokat. Konfigurációs beállításokat is lehet visszaküldeni.
4. A `leap.js` kliens JavaScript könyvtár használható a webes alkalmazásokhoz. A könyvtár felépíti a kapcsolatot és fogadja a JSON üzeneteket. Az API felépítése és strukturája hasonló a natív API-hoz.

IP Kamera

típusa: FI9826W

A kamera főbb adatai:

felbontása: 1280 x 960 pixel max. 30 fps

Lencse: f: 4-9mm, F1.8

Tömörítés: H. 264 (videó), PCM/G.726 (audió)

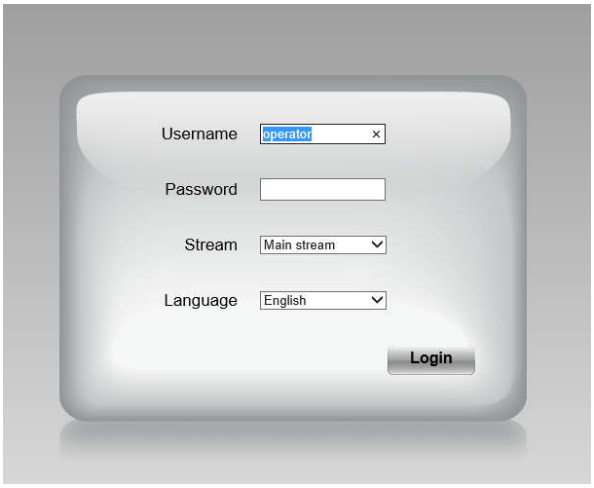
Mozgathatóság: vízszintes irányban: 300°

függőleges irányban: 120°

Vezérlése:

Webes felületen keresztül:

Bejelentkezés:



A felületen megtalálható irányító gombokkal a kamera mozgatható. A „zoom in” és a „zoom out” gombokkal a nagyítás mértéke változtatható

RTSP kapcsolat

A videó kamera streamjét RTSP protokollon keresztül is lehet vezérelni. A RTSP egy HTTP-hez hasonló felépítésű protokoll.

RTSP: Real Time Streaming Protocol

A protokoll nem közvetlenül a tartalom átviteléért felelős, hanem csak az irányításért. Például a lejátszás indítása, szüneteltetésére, vagy a stream egyes paramétereinek megismerésére használható. A tartalom átvitele többnyire RTP (Real-time Transport Protocol) protokollon keresztül történik.

Kamera API:

A kamera HTTP-n keresztül érhető el, általános formában a következő módon:

http://w.x.y.z/cgi-bin/CGIProxy.fcgi&usr=username&pwd=password&cmd=ABC

Eredmény XML formában:

```
pl: <CGI_Result>
    <result>0</result>
    <resolution0>6</resolution0>
    <resolution1>1</resolution1>
    <resolution2>1</resolution2>
    <resolution3>1</resolution3>
    <bitRate0>4194304</bitRate0>
    <bitRate1>2097152</bitRate1>
    <bitRate2>1048576</bitRate2>
    <bitRate3>204800</bitRate3>
    <frameRate0>10</frameRate0>
    <frameRate1>25</frameRate1>
    <frameRate2>15</frameRate2>
    <frameRate3>10</frameRate3>
    <GOP0>10</GOP0>
    <GOP1>30</GOP1>
    <GOP2>45</GOP2>
    <GOP3>60</GOP3>
    <isVBR0>1</isVBR0>
    <isVBR1>1</isVBR1>
    <isVBR2>0</isVBR2>
    <isVBR3>0</isVBR3>
</CGI_Result>
```

Ahol: result mező értelmezése:

<result></result> means the common execute result

value	mean
0	Success
-1	CGI request string format error
-2	Username or password error
-3	Access deny
-4	CGI execute fail
-5	Timeout
-6	Reserve
-7	Unknown error
-8	Reserve

Kamera mozgatása:

A kamera mozgatása két lépésből áll. Az első utasítással el kell indítani a kamera mozgatását. A kamera ezek után addig mozog, amíg le nem éri valamelyik végállását, vagy le nem állítjuk a ptzStopRun paranccsal.

/cgi-bin/CGIProxy.fcgi?cmd=ptzStopRun

/cgi-bin/CGIProxy.fcgi?cmd=ptzMoveUp

/cgi-bin/CGIProxy.fcgi?cmd=ptzMoveDown

/cgi-bin/CGIProxy.fcgi?cmd=ptzMoveLeft

/cgi-bin/CGIProxy.fcgi?cmd=ptzMoveRight

/cgi-bin/CGIProxy.fcgi?cmd=ptzMoveTopLeft

/cgi-bin/CGIProxy.fcgi?cmd=ptzMoveTopRight

/cgi-bin/CGIProxy.fcgi?cmd=ptzMoveBottomLeft

/cgi-bin/CGIProxy.fcgi?cmd=ptzMoveBottomRight

A kamera nagyítása is állítható (zoom). A működés hasonló a kamera mozgatásához, a zoom elindítása után a zoom a végállásig megy vagy a zoomStop utasítással megállítható.

/cgi-bin/CGIProxy.fcgi?cmd=zoomStop

/cgi-bin/CGIProxy.fcgi?cmd=zoomOut

/cgi-bin/CGIProxy.fcgi?cmd=zoomIn

A kamera alaphelyzetbe állítása:

/cgi-bin/CGIProxy.fcgi?cmd=ptzReset

A kamera mozgatási sebességének lekérdezése:

/cgi-bin/CGIProxy.fcgi?cmd=getPTZSpeed

A kamera előre beállított pozicióinak lekérdezése:

/cgi-bin/CGIProxy.fcgi?cmd=getPTZPresetPointList

/cgi-bin/CGIProxy.fcgi?cmd=ptzDeletePresetPoint&name=test

/cgi-bin/CGIProxy.fcgi?cmd=ptzAddPresetPoint&name=test

/cgi-bin/CGIProxy.fcgi?cmd=ptzGotoPresetPoint&name=Alap

Mérési Feladatok:

(F.1) Indítsd el a Leap Motion Diagnostic Visualizer-t

tipp: jobb alsó sarokban, kisméretű leap motion ikonra jobb egérrel rákattintva Visualizer…

Ellenőrizd, hogy kezedet az eszköz fölé tartva követi-e a rendszer.

Állítsd teljes képernyőre az alkalmazást (s)

Kapcsold ki a kameraképet (f)

Jelenítsd meg a help funkciót és sebességi adatoka (h)

Figyeld meg az eszköz sebességét! Bal felső sarokban (device fps). Milyen értékek között változik?

Jegyezd fel a jegyzőkönyvbe az értékeket! Mentsd el a képernyőképet is!

tipp: Teljes képernyős módban nem tudod elmenteni a képet, válts vissza ablakos módba (s). Alt-PrtSc –vel csak az ablak képét menti a vágólapra.

Forgasd el a 3D-s megjelenítést (→ ←)

Próbáld ki a többi funkciót is!

Mentsd el a képernyőképet!

Tartsd mindkét kezed a Leap Motion eszköz fölé! Keress olyan poziciót, amikor mindkét kezed az eszköz felett van, de mégsem jól ismeri fel!

Kamera:

(F.2) Jelentkezz be a webes felületen Internet explorerből!

http://w.x.y.z:88 (A mérőcsoporthoz tartozó ip címet a mérésvezető adja meg)

Tanusítvány hibát fog jelezni, tovább kell lépni!

„Kívánja kizárólag a biztonságosan kézbesített tartalmakat megjeleníteni? Válasz: NEM

felhasználónév: operator

jelszó: oper

Mozgasd a kamerát a bal oldali vezérlő segítségével! Állítsd középre a zöld növényt és nagyíts bele. Mentsd el a képet!

Nézd meg a mennyezeti lámpa oldalát a kamerával, ha a fali kamerát kezeled! Keress egy 6 jegyű azonosítót!

Ha rack szekrény tetején lévőt használod, a rack szekrény belső oldalán keresd a 6 jegyű azonosítót! Olvasd le a 6 jegyű azonosítót! Mentsd el a képet!

Ip stream megfigyelése:

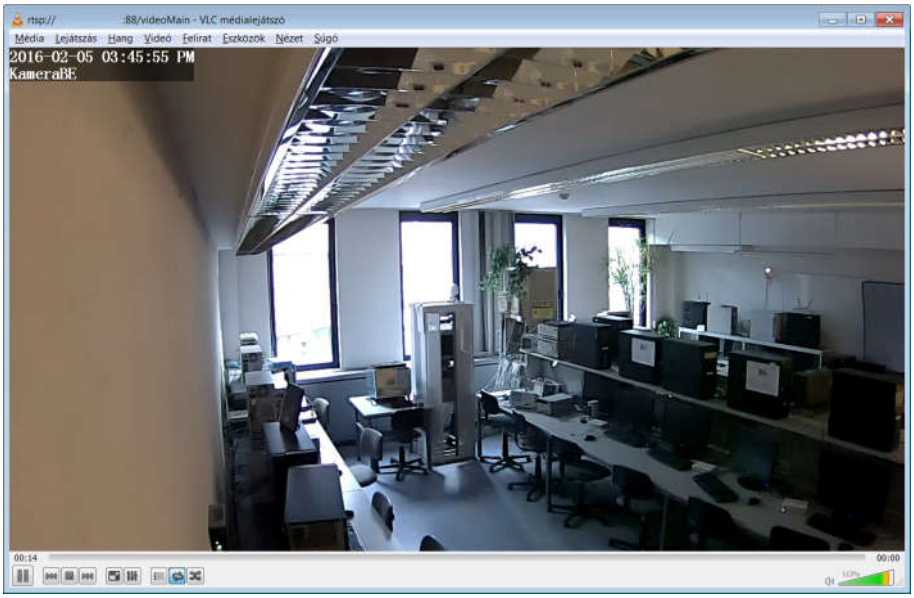
indítsd el a VLC média player-t

Menüben: Média/Hálózati műsor megnyitása:

rtsp://w.x.y.z:88/videoMain (A mérőcsoporthoz tartozó ip címet a mérésvezető adja meg)

Felhasználó név/jelszó: visitor/visi

Mentsd el a képernyőképet:



Feladat:

(F.3) VLC playerrel figyelni a stream-et, közben böngészőből API-val irányítani a kamerát!

Figyelem! Nem a kamera saját felületén kell mozgatni a kamerát, hanem HTTP hívásokkal!

Először kérdezze le és értelmezze a zoom sebességét:

http://w.x.y.z:88/cgi-bin/CGIProxy.fcgi?cmd=getZoomSpeed&usr=operator&pwd=oper

(A mérőcsoporthoz tartozó ip címet a mérésvezető adja meg)

0- Gyors
1- Normál
2- Lassú

Rögzíts a sebességet a jegyzőkönyvben!

Mozgasd a kamerát, hogy az ajtó látszódjon!

Mentsd le jegyzőkönyvbe a képernyőképet és a felhasznált parancsokat!

tipp: Ha nem reagál a http kérésre, akkor a Reload gomb megnyomása szükséges, mert a böngésző cache-ből adja a választ, nem kapja meg a kamera a kérést.

Számítsd ki a vízszintes forgatás sebességét (szög/mp)-ben!

Hello world program

(F.4) A programozáshoz a Visual Studiót használd! A .sln fájlt nyisd meg!

A Leap Motion API-hoz segítséget találasz a D:\tmp\docs könyvtárban.

Másold a D:\tmp\helloworld könyvtárban található sample.cpp fájlba a következő kódot:

```
#include <iostream>
#include <string.h>
#include "Leap.h"

using namespace Leap;

int main(int argc, char** argv) {

    // Keep this process running until Enter is pressed
    std::cout << "Press Enter to quit..." << std::endl;
    std::cin.get();

    return 0;
}
```

Fordítsd le, futtasd le! Ellenőrizd a működését!

Kapcsolat ellenőrzése:

A kapcsolat a Leap Motion szolgáltatással a `Controller` objektumon keresztül zajlik. Módosítsd a programot:

```
int main(int argc, char** argv) {
    Controller controller;

    // Keep this process running until Enter is pressed
    std::cout << "Press Enter to quit..." << std::endl;
    std::cin.get();

    return 0;
}
```

Fordítsd le, futtasd le! Ellenőrizd a működését!

Bővítsd a programot egy Listener objektummal, amely segítségével lehet adatokat kapni a szolgáltatástól. Bővítsd a kódot:

```
class SampleListener : public Listener {
    public:
    virtual void onConnect(const Controller&);
    virtual void onFrame(const Controller&);
};
void SampleListener::onConnect(const Controller& controller) {
    std::cout << "Connected" << std::endl;
}
void SampleListener::onFrame(const Controller& controller) {
    std::cout << "Frame available" << std::endl;
}
```

Módosítsd a main függvényt, amelynél a Listener objektumot a Controller-hez rendelheted:

```
int main(int argc, char** argv) {
    SampleListener listener;
    Controller controller;

    controller.addListener(listener);

    // Keep this process running until Enter is pressed
    std::cout << "Press Enter to quit..." << std::endl;
    std::cin.get();

    // Remove the sample listener when done
    controller.removeListener(listener);

    return 0;
}
```

Fordítsd le, futtasd le! Ellenőrizd a működését!

Alapértelmezetten a gesztusokat nem ismeri fel, ezeket engedélyezni kell. Módosítsd a programot:

```
void SampleListener::onConnect(const Controller& controller) {
    std::cout << "Connected" << std::endl;
    controller.enableGesture(Gesture::TYPE_SWIPE);
}
```

Fordítsd le, futtasd le! Ellenőrizd a működését!

Az adatokat Frame-ek formájában küldi a szolgáltatás. Módosítsd a programot, hogy a Frame legfontosabb adatait kiírja a képernyőre a program:

```
void SampleListener::onFrame(const Controller& controller) {
    const Frame frame = controller.frame();
    std::cout << "Frame id: " << frame.id()
          << ", timestamp: " << frame.timestamp()
          << ", hands: " << frame.hands().count()
          << ", fingers: " << frame.fingers().count()
          << ", tools: " << frame.tools().count()
          << ", gestures: " << frame.gestures().count() << std::endl;
}
```

Fordítsd le, futtasd le! Ellenőrizd a működését!

(F.5) Módosítsd a programot, ha Swipe gesztust ismer fel a rendszer, lekérje az alpha.tmit.bme.hu honlap tartalmát. Ehhez használd a következő fájlokat:

A HTML hívásokhoz rendelkezésre áll a cURL lib és egy abból összerakott osztály.

Az osztályból létre kell hozni egy példányt pl:

```
#include "CurlEasyGet.h"
CurlEasyGet mycurl;
```

A get tagfüggvényével lehet egy HTML hívást végrehajtani:

```
mycurl.get("http://alpha.tmit.bme.hu/");
```

Az eredmény a mycurl.mem struktúrában található:

```
printf("%i bytes: %s\n",mycurl.mem.size,mycurl.mem.memory);
```

Feladat!

Mozgasd a kamerát, hogy az ajtó látszódjon! Közelíts rá! Használj előre tárolt pozíciót kiindulási állapotként, például az „Alap” nevű előre eltároltat. Végállapot nem lehet tárolt pozíció! Nem kell képfelismerést alkalmazni, csak relatív mozgásokat. A program írja ki, hogy mit csinál éppen. Az időzítési adatokat kísérletezéssel kell megállapítani.

Mentsd le jegyzőkönyvbe a képernyőképet!

Figyelem! A programnak minden esetben tudnia kell produkálni a kívánt képernyőképet!

mj: Időzítéshez használd a Sleep() függvényt! A paraméterként ms-ben kell a várakozási időt megadni:

```
#include "Windows.h"


Sleep(100);
```

Gesztusvezérlés program:

A D:\tmp\Gestusdemo könyvtrában található programmal dolgozz!

(F.6)Készíts c++-ban programot amely:

- Tervezz gesztus alapú felhasználói interfészt, amely segítségével a kamera vezérelhető.
- Vedd figyelembe az ember-gép interfész órán tanultakat!
- A kamera minden irányba (fel, le, jobbra balra, nagyítás, kicsinyítés) mozgatható legyen leap motion segítségével. Javasoljuk, hogy használd a tenyér pozícióját vagy dőlésszögét a vezérléshez.
- Legyen lehetőség előre eltárolt pozíciók kiválasztására.
- Nyomkövető információk jelenjenek meg a konzolos program képernyőjén. A program csak a releváns információkat írja ki, szemmel jól követhető és olvashatók legyenek a megjelenített információk.
- A kamerát a cgiProxy-n keresztüli http hívásokkal kell vezérelni
- A működéshez készíts felhasználói utasítást, amely leírja a használatot.
- A programból a felesleges részeket tedd megjegyzésbe, hogy gyorsabb legyen a program, és a képernyőn is csak a hasznos információk jelenjenek meg!

Irodalomjegyzék:

[1] https://developer.leapmotion.com/documentation/cpp/devguide/Leap_Overview.html

[2] http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/

[3] https://developer.leapmotion.com/getting-started/javascript

Jegyzőkönyv

- Figyelem! A mérési jegyzőkönyv alapján a mérést meg kell tudni ismételni, és az eredményeket a legpontosabban reprodukálni lehessen. A jegyzőkönyvben ne szerepeljenek a mérési utasítások, illetve szükségtelen egyéb információk. A jegyzőkönyvben használd a feladatok sorszámát pl:(F.1). A mérés során elvégzett feladatok pontos leírása, az eredmények dokumentálása. Az elkészült programokat a jegyzőkönyv végén kell mellékletként beadni.
- A mérés elvégzéséhez kapcsolódó megjegyzések, szubjektív vélemények.

  A mérési jegyzőkönyv mellékleteként beadandó: A mérés során készített programok forrása. A forrás bemásolásakor egyértelműen ki kell derülnie, hogy hol kezdődik a fájl, hol végződik és mi volt a fájl neve.

Jegyzőkönyv

VITMMB04 Smart City Laboratory, Gesture Control

Mérés helye:
Ideje:
Mérést végző hallgatók neve és neptun kódja:

The equipment's identifiers:

Számítógép neve:
Számítógép IP címe:
Operációs rendszer típusa és verziója:
Leap Motion Id (1,2 or 3):
Leap Motion szoftver verziója:
Leap Motion Controller ID:
Leap Motion Firmware Revision:
Webkamera típusa:
Webkamera IP címe:
VLC player verziója:
Web böngésző típusa és verziója:
Visual Stúdió verziója:

1. Feladat:

2. Feladat:

3. Feladat:

4. Feladat:

5. Feladat:

6. Feladat:

Egyéb észrevételek, megjegyzések, a méréshez kapcsolódó vélemények:

Csatolmányok:

4. feladat forrása (csak a cpp fájlok)
5. feladat forrása (csak a cpp fájlok)
6. feladat forrása (csak a cpp fájlok)

A mérési jegyzőkönyvet és a forrás fájlokat egy zip fájl formájában kell feltölteni a következő oldalon keresztül:

http://smartlab.tmit.bme.hu/education-OV-upload_login

Beugró kérdések:

1. Mire való az RTSP protokoll?
2. Mi a Leap Motion eszköz működési elve?
3. Soroljon fel legalább 3 gesztust, amit a Leap Motion API alapértelmezetten fel tud ismerni!
4. Soroljon fel legalább három objektumot, ami a Leap Motion API alapértelmezetten kezel!
5. A webkamera CGI hívása milyen formában adja vissza az eredményt?
6. Milyen parancsokkal lehet a kamerát mozgatni? Adjon legalább 3 példát!
7. Mi a Leap Motion két fajta alap programozó interfésze?
8. Mi a funkciója a Leap Motion API-ban az onFrame függvénynek?
9. Mit csinál a controller.enableGesture(Gersure::TYPE_SWIPE)? Miért kell meghívni?
10. Mit azonosít a Leap Motion eszközként?