ATLAS: Virtual Presence and Campus Map Robot

Kevin Cox

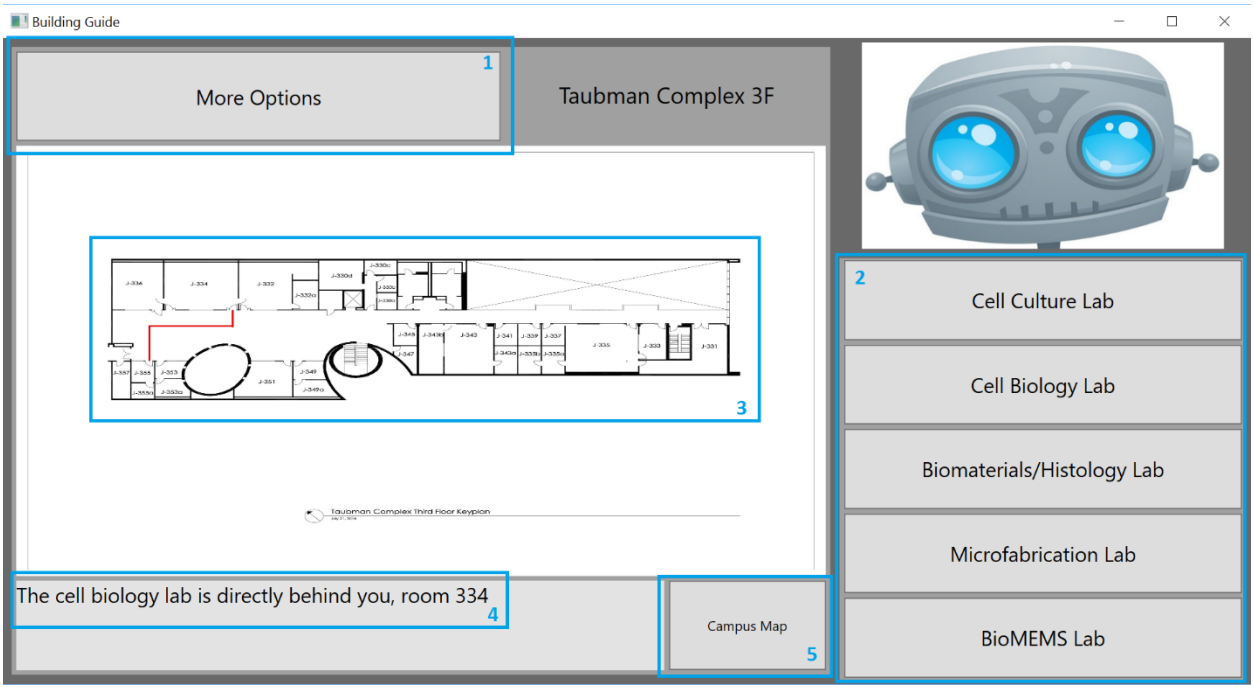
Purpose:

The purpose of this robot is to provide directions to certain rooms and buildings on campus. The robot also has the capability to act as a mobile virtual presence device. The robot has an autonomous mode that allows the robot to turn to and face a direction to emphasize its directions to a user. Also, the robot has a manual mode where a remote operator can control the robot's movements. These two modes are mutually exclusive to prevent the remote operator and user from controlling the robot's movements at the same time. The robot gives directions to a user using Microsoft's .NET framework's voice synthesizer. The UDP video streaming is based on Khalil Khalaf's Java code for his MP-ConBot project.

Project Code Repository: https://github.com/Kebrador/Autonomous-Map-Robot

GUI Overview:

Building Guide GUI (Robot side)



1. More options button:

This button lists all the available locations the robot can guide a user towards.

2. Quick buttons:

These buttons offer the user a short list of locations the robot can guide them towards.

3. Map:

The map shows the user a line to guide them to a location. The map can also be used to select locations directly.

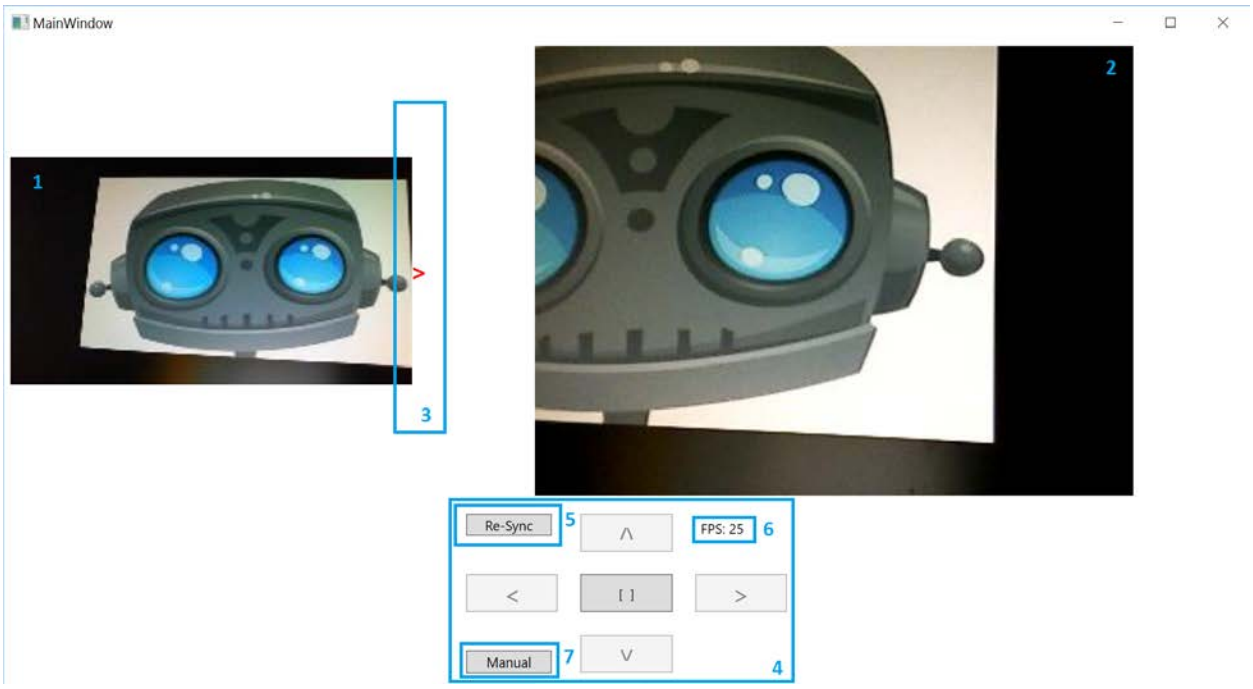
4. Subtitle:

This area displays the subtitle for the robot's speech synthesizer.

5. Map switch

This button switches between the campus map and the Taubman Complex 3rd floor map.

Controller GUI (Client side):



1. Upper Image (image shown not from actual configuration)

In actual configuration, the upper image is streamed from the built-in camera on the Surface Pro mounted at eye level.

2. Lower Image (image shown not from actual configuration)

In actual configuration, the lower image is streamed from a wide-angle camera mounted on the base of the robot.

3. Image focus switch:

This button allows the user to switch which image is displayed larger on the GUI.

4. Robot controls:

This includes the stop and directional buttons. The robot can also be driven with the WASD or arrow keys. Pressing any other button will cause the robot to stop. Other control buttons are listed separately.

5. Start Video/Re-Sync button:

This button starts the video stream from the robot. This button can also be used to re-sync video with the robot if the video is interrupted for any reason.

6. FPS counter:

Shows the average FPS between the two video streams.

7. Mode switch button:

This button allows the user to switch between manual and autonomous modes. In autonomous mode, the robot will move interactively based on user input and cannot be controlled remotely. In manual mode only the user can make the robot move.

Functionality:

Key elements of the robot's functionality include multiple camera streaming, randomized idle actions, autonomous and manual modes, video timeout and resync, keyboard based controls and an interactive map.

1. Multiple camera streaming:

Cameras are separated into their own threads and are based on the same method.

```
static void Camera1()
{
	camthread(0);
}

static void camthread(int camera_number)
{
	Capture cap = captures[camera_number];
	…
```

Each camera thread has its own camera capture (selected by the camera_number) and its own UDP socket. The port number for each camera is different and is determined by 15001 + camera_number. The UDP socket will wait to receive a message from the controller and sends the picture over UDP via bitmap. There is no FPS cap and the actual FPS stays between 14 and 25 FPS in normal operation. The image is resized before being sent to optimize FPS and prevent the video stream from becoming corrupted.

2. Randomized idle actions:

If the robot is in autonomous mode and has not received input for two minutes it will enter idle mode and randomly ask if it can assist anyone with directions. The minimum time between random talking is 3 minutes. After 3 minutes the chances of the robot talking increases every second to a maximum of 8 minutes.

3. Autonomous/Manual modes:

The robot can be switched between manual and autonomous modes by the remote user. The default mode is autonomous. The differences between autonomous and manual modes are detailed in the GUI Overview section. The mode switch message is the only message that requires a return confirmation message.

```
static void autonomous_mode_start()
{
	MyBot.stop();
	byte[] data = Encoding.ASCII.GetBytes("Confirm");
	MainServerSocket.Send(data, data.Length, MainClient);
	autonomous = true;
	idle_count = 0;
}
```

4. Video stream time-out:

The client side video request thread has a time-out feature that helps to prevent video freezing.

```
private void vid_thread1()
{
	while (true)
	{
		var task1 = Task.Run(() => vid_rec(0, main_image));
		if (!task1.Wait(TimeSpan.FromMilliseconds(250)))
		{
			failcount1++;
			if (fail_sock[0] != null)
				fail_sock[0].Close();
		}
		framecount1++;
	}
}
```

During timeout, the socket of the video thread needs to be closed in order to create a new socket for the next frame.

5. Keyboard based controls:

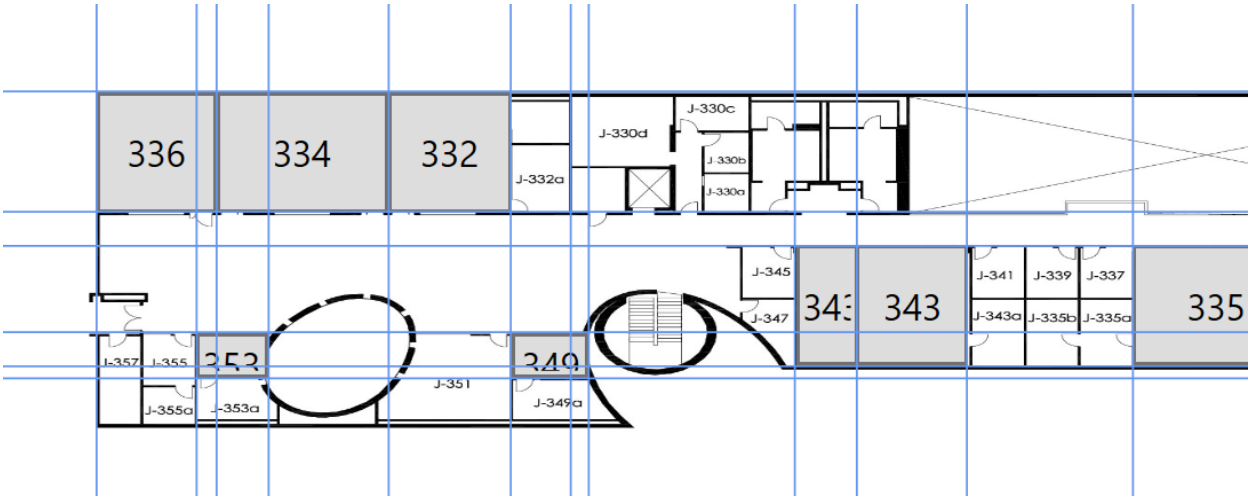
The robot can be remotely operated with the arrow keys and WASD keys as well as the buttons on the GUI.

```
private void Window_KeyDown(object sender, KeyEventArgs e)
{
    if (m_override)
    {
```

```
        if (e.Key == Key.Up || e.Key == Key.W)
            control_l2bot("Forward");
        else if (e.Key == Key.Down || e.Key == Key.S)
            control_l2bot("Backward");
        else if (e.Key == Key.Left || e.Key == Key.A)
            control_l2bot("Left");
        else if (e.Key == Key.Right || e.Key == Key.D)
            control_l2bot("Right");
        else
            control_l2bot("Stop");
    }
}
```

6. Interactive Map:

The map is covered in hidden buttons that have the same functionality as pressing any of the other dedicated buttons. These buttons are invisible to the user, but can be interacted with by touching the appropriate room on the map.



Acknowledgement

The physical robot platform based on L2Bot architecture was provided by Prof. CJ Chung. CS robotics lab assistant Devson Butani modified L2Bot and constructed the upper part of the robot.