TelSurge Documentation

Table of contents

Introduction 3
System requirements 4
Quick Start Guide 4
System Architecture 5

Introduction

TelSurge

TelSurge is a system created for the TATRC project and provides telesurgical and telestrative features to a surgical robotic system. It consists of a network of locally (or VPN) connected users. Users are sorted into 2 types: Master or Client. For a given surgery, there is one Master that is in the operating environment and is physically connected to the robot. There can be any number of Clients connected to the same Master. Thus, this system resembles a star typography (Fig. 1).

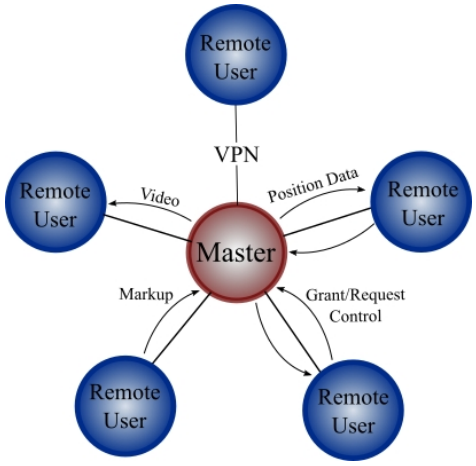

Figure 1. TelSurge represented as a star typography

This system allows for multiple surgeons to share control of a surgical robot by handing off control back and forth. This process is described in Fig. 2.

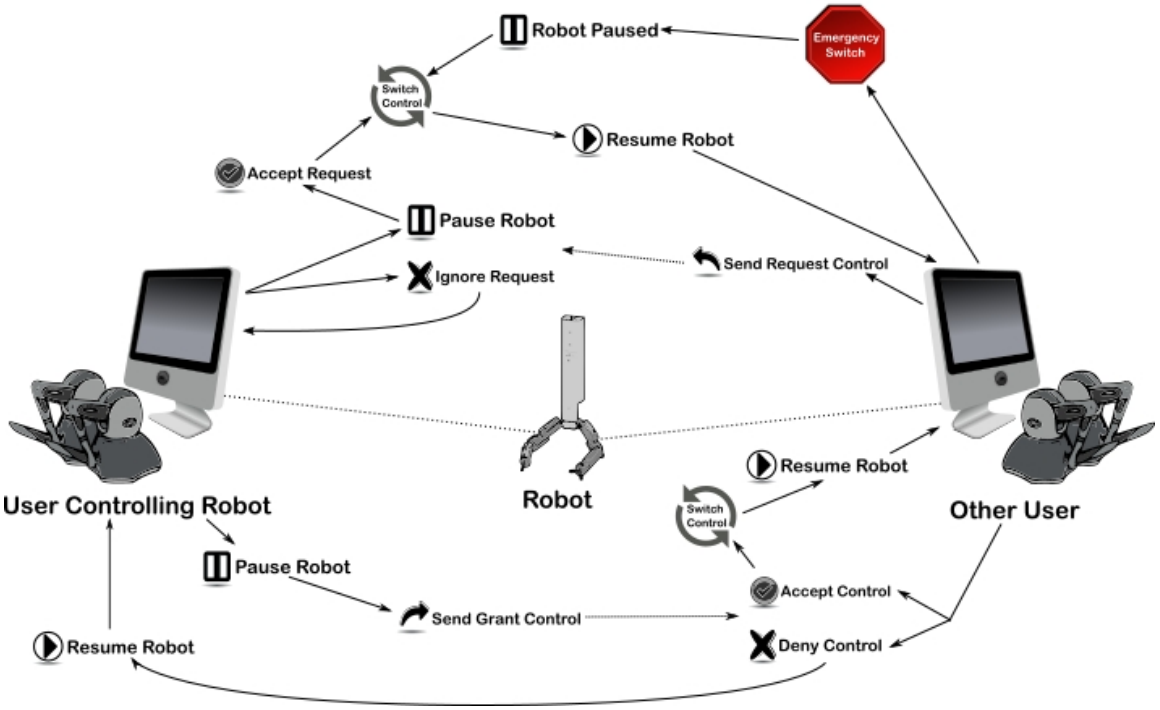

Figure 2. Sequence of switching control of a robot

This system also allows for telestration, or markup, overlaid on a video and shared between all users. An example is shown on Fig. 3.

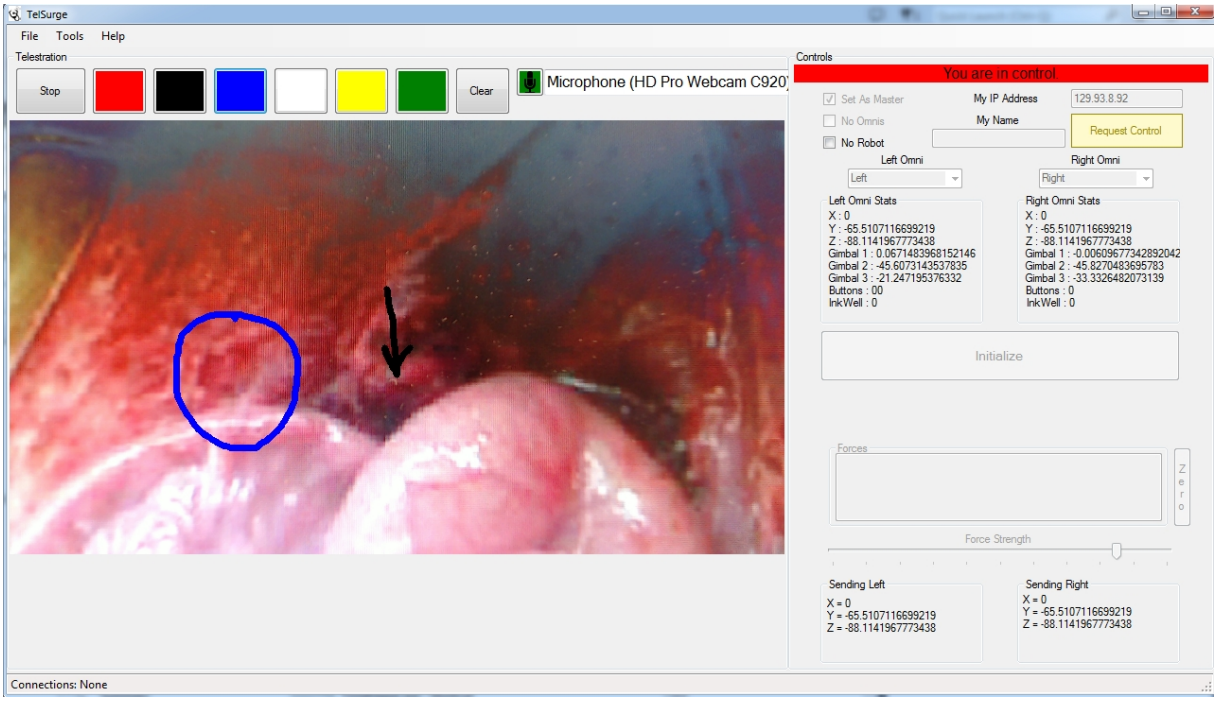

Figure 3. Example of TelSurge using telestration feature

For an in-depth look at the source files of TelSurge, take a look at the System Architecture section of this document.

System requirements

System Requirements

TelSurge requires the following to run on a machine:

- Windows (7 or up)
- Processor (Core 2 Duo or better)
- Internet connection with VPN capabilities
- Geomagic Touch Omni controllers (optional, required for controlling surgical robot)
- Touch Screen (optional)

The TelSurge system also has some software dependencies which include:

- Geomagic Touch device drivers
- EmguCV - C# wrapper for OpenCV
- DirectShow - for video capture
- Newtonsoft - for JSON serialization

Quick Start Guide

Quick Start Guide

GitHub Repository

TelSurge is part of the surgical-robots/robot-control-app GitHub repository. To get a copy of TelSurge, you will need to download the source code and run it through Visual Studio as there is no official release of the system.

Starting Up

Master (connected to surgical robot)

To run as Master, you must have Geomagic Touch Omnis connected to your machine.

1. Start TelSurge
2. Check "Set As Master" box
3. Select connected Omnis in the drop down boxes (Left and Right)
4. Click Initialize

Client

1. Start TelSurge
2. If you do not have Omni controllers, check "No Omnis" box
3. Input the Master's IP address in the "Master's IP Address" box
4. Input your name in the "My Name" box
5. If you do have Omnis, select your left and right controllers in the drop down boxes
6. Click Initialize
7. Click connect to Master

Configuration

These are ways that you can configure your TelSurge session.

- Change what IP address from your machine that TelSurge uses to communicate with by selecting Tools / Change My IP.
- Join or leave the TelSurge audio conference by clicking the green or red microphone button.
- Select which microphone to use with the audio conference by selecting it from the drop down box to the right of the microphone button.
- Draw telestration shapes on the video feed by selecting your color and clicking and dragging over the video viewer.
- Request control of the surgical robot by clicking the yellow "Request Control" button in the top right.
- Grant / request control of the surgical robot by clicking available names of users in the status bar and the bottom left of the main window.

System Architecture

System Architecture

Organization

TelSurge source code is part of the RobotControl solution found in the surgical-robots/robot-control-app GitHub repository. Inside the TelSurge project, source files are sorted into classes and windows forms. Forms contain both front-end (graphical) and code-behind (C#) components. Forms, such as TelSurgeMain.cs, are located in the root directory of the TelSurge project. Classes on the other hand, are found in the DataModels folder.

Windows Forms

TelSurgeMain.cs

TelSurgeMain.cs is the root of the system. It is a windows form that contains the main UI (User Interface) and the main thread of the system. The form is universal and all types of users (Master, Client) run the same form.

Functions

1. Runs the main thread of TelSurge
2. Spawns all subsequent threads
3. Serves as a central hub for all communication processes
4. Contains the main GUI (Graphics User Interface)

AssignButtons.cs

This form allows a user to map specific hardware buttons to certain TelSurge events. Currently, there are three events to map to:

1. Emergency Switch

   The emergency switch is used in the case of the local surgeon needing to gain control of the surgical robot right away and thus bypasses the "handoff" that is usually done before switching control. Only the local surgeon has this capability and when it is pressed, control of the robot is immediately given to the local surgeon while "freezing" or pausing the robot.

2. Freeze Button

   The freeze button is used for pausing, or "freezing" the robot. When the robot is frozen, movement of the controllers (Geomagic Touch Omnis) does not affect the robot. When the freeze button is pushed again, the freeze is disengaged and the controllers are able to move the robot.

3. Following Button

   The following button is used primarily for training scenarios and allows a user to feel the controller movements of the surgeon in control of the robot. This process is called following because the user's controls "follow" the surgeon's controls.

Functions

1. Map connected hardware buttons to key TelSurge events
2. Allows the user to chose an option from a list of available buttons

CameraControl.cs

This form controls a connected PTZ (Pan-Tilt-Zoom) camera and attached laser pointer. It connects to a PTZOptics camera and sends a HTTP request for all movements (Up, Down, ZoomIn, etc.). Although it is setup to work with this specific brand of camera, many PTZ cameras allow for similar control through HTTP requests or TCP packets. The laser pointer button connects to a Photon, a wifi-enabled arduino-like chip from Particle and through a Particle cloud service, sends a command to the photon to set a pin HIGH and power the laser pointer.

Functions

1. Control a PTZ camera through TelSurge
2. Send a message to a microcontroller to turn on a laser pointer mounted on top of the camera.

ChangeMyIP.cs

TelSurge requires that all users are connected to the same local network (or via VPN) to communicate. However, since a computer may have several different IP addresses, it is required to specify which address the user wants to be bound to for communication. This form allows a user to choose which IP address on which to listen for other TelSurge connections. The chosen address should be the one associated with the

local network shared with other TelSurge users. This should be done before connecting to the Master, or before initializing.

Functions

1. To allow a user to choose which IP address to use for TelSurge communication.

ChangeVideoSource.cs

This form allows a user to select which video feed it wants to be displayed. The Master can choose any connected camera or video source or a network video source. The Master sends whatever source it chooses to all Clients as a Master video feed. A Client can choose any connected camera, video source, network source, or the video feed it receives from the Master. The list of available options is a combination of:

1. All locally connected video sources / cameras
2. The Master feed if the user is a Client
3. All network sources that are specified in the /Content/InternetCameras.xml file

This file may be manually updated or updated through TelSurge by selecting: Tools / Video Source / Add IP Camera.

Functions

1. To give a user the ability to select a video source for the main TelSurge viewer (TelSurgeMain)

ConnectButtons.cs

This form is used to connect foot pedals to TelSurge. It works specifically with a pedal array controlled by a Teensy (Arduino-like) micro controller. In this form, a user selects a COM port that corresponds to the Teensy and connects to it. This form must stay open throughout the running of the TelSurge app if using pedals because it is actively communicating with the micro controller.

Functions

1. To connect a Teensy - controlled foot pedal array.
2. Constantly communicates with the Teensy micro controller to update TelSurge with the state of the external buttons (pedals).

IPCameras.cs

This form is used to modify the /Content/InternetCameras.xml file. This file contains the connection details of network video sources like IP cameras.

Functions

1. Allows a user to save a network video source or camera URL and connection information to use later when selecting a video source to view. The ChangeVideoSource.cs form pulls from this configuration file.

NetDelay.cs

NetDelay is a form used to artificially cause a delay, or lag in TelSurge communication. A user enables this delay by selecting *Set Delay* on the form. TelSurge will delay the video feed and controller (Omni) information it receives for the amount of time selected by the numeric scroll on the form. This form was created to test the difficulty of robotic telesurgery under poor network speeds.

Functions

1. Delay the user's incoming TelSurge communication data from being used for a certain amount of time.

Classes

AudioConference.cs

This class handles all audio communication between connected TelSurge users. It creates an AudioConference object and allows a user to join and leave a conference. It also handles sending and receiving audio packets. This class is ran as a separate thread and spawned from TelSurgeMain.cs.

Functions

1. Define an AudioConference object.
2. Allow a user to join and leave an audio conference.
3. Handle sending and receiving audio packets.

Figure.cs

Figure.cs is a simple structure for the telestration features of TelSurge. A figure is defined when a user clicks and holds over the video viewer of TelSurgeMain.cs until the mouse is release. For example, if the user were to draw a triangle in Microsoft Paint, the user would select a tool such as the pencil and click and drag in a triangular shape. In this case, the triangle would be a figure. A figure object has two properties:

1. Color: the color of the figure. A figure can only have one color.
2. Array of Points: an array of pixel coordinates that make up the shape.

Functions

1. Define a structure that contains the information of a mouse-drawn shape.

IPCamera.cs

This is a structure that defines a IPCamera object that is used in the IPCameras.cs form and the /Content/InternetCameras.xml file. This object contains four properties:

1. Identifier: unique integer assigned for each IP camera
2. Name: the recognized name for the IP camera
3. Address: the string representation of the URL or IP address of the camera
4. PTZAddress: the URL for where to send Pan, Tilt, Zoom commands over HTTP requests (optional)

Functions

1. Define a structure that contains the information needed to connect to a network video source or IP camera.

Location.cs

This is a very simple structure that is only used by the NetDelay.cs form to allow a user to select a specific location to model a connection with in terms of connection speed.

Functions

1. Define a structure that contains a location and its distance from Lincoln, NE.

Logging.cs

This is a class that is used to define and write to a system log represented by a CSV file located in the run directory (ex. /bin/Debug). This is a centralized way to record run performance and errors discretely while running TelSurge. Errors can be displayed on the TelSurgeMain.cs form and recorded or just recorded directly to the log as in a silent error. This object has four properties:

1. StatusMessage: the short main message to add to the log

2. StatusDetail: An extended message giving more details about the event
3. StatusType: [Error, Running, or Stopped] based on status of the program at the time of the event
4. URL: The location of the CSV log file to be used

Functions

1. Define a structure for logging system events and errors
2. Write a message to a CSV log file

Markings.cs

This is a structure that represents the visual telestration markings of a user. It consists of a list of Figure.cs objects for each color TelSurge supports: Red, Black, Blue, White, Yellow, and Green. It also has OffsetX and OffsetY properties that are used when scaling the video viewer of TelSurgeMain.cs (offsets not currently implemented). This class also has functions:

- Merge: allows a Markings.cs object to be merged with another Markings.cs object
- GetAllPaths: returns the paths of all the markings of a specific List of Figure.cs objects
- Clear: removes all markings from a Markings.cs object
- RemoveLastFigure: removes the last Figure.cs object from a specified List

Functions

1. Define an object that contains all telestration shapes of all colors for a user
2. Allow a user to manipulate a Markings.cs object

Markup.cs

This class handles the telestration feature of TelSurge. It is ran as a separate thread from the TelSurgeMain.cs form. It manages all telestration from a user and handles communicating the telestration to other users. All users send their telestrations to the Master. Then the Master merges them with its own. Then all the telestration is embedded in the Master's video feed and sent to all Clients. Telestration only works when the client's video source is set to Master's video feed.

Functions

1. Defines a class that handles telestration for a user
2. Handles communication of telestration between users
3. Allows for clearing of all user telestration

OmniPosition.cs

This is a structure for controller information coming from a Geomagic Touch Omni as well as any extra buttons (foot pedals). OmniPositions can be added and subtracted from each other which is used in a "clutch" type of position offsetting.

Functions

1. Define an object that contains Geomagic Touch Omni position information and any foot pedal information
2. Allow positions to be added and subtracted from eachother

SocketData.cs

This class handles standard messages between a user and the Master. This class runs as its own thread spawned from TelSurgeMain.cs. It handles both UDP and TCP type protocols for sending data packets. Also, it handles serializing and deserializing objects in and out of JSON.

Functions

1. Defines a class that handles UDP/TCP communication of objects between users
2. Serializes an object into JSON
3. Deserializes JSON into an object
4. Sends and receives local network packets

SocketMessage.cs

This is a structure that contains the TelSurge data shared between a user and the Master. It has nine properties:

1. OmniPosition: OmniPosition.cs object that contains the position of two Geomagic Touch Omni controllers
2. PositionOffset: OmniPositino.cs object that represents the offset of the controllers for the "clutch" feature when controlling the surgical robot.
3. User: the User.cs object representing the current user
4. Forces: the OmniPosition.cs object that contains the amount of haptic forces that should be exerted using the Geomagic Touch Omni controllers
5. sendToggleFrozen: boolean representing whether or not a "Frozen" state should be toggled at the RobotControl end of the software solution
6. EmergencySwitch: boolean representing the state of the Emergency Switch that swiftly gives control to the local surgeon
7. ClearMarkingsReq: boolean telling the recipient of the message to clear their telestration.
8. Port: the port to send this message to
9. Surgery: the Surgery.cs object the represents the TelSurge connected network of users

Functions

1. Defines a structure for TelSurge data that is sent between users

Surgery.cs

This is a structure that represents a network of connected TelSurge users all participating on the same surgery. This class contains four properties:

1. InControlPosition: OmniPosition.cs object the contains the Omni controller position data for the user that is currently in control of the surgical robot
2. Master: User.cs object that represents the Master of the surgery
3. ConnectedClients: the list of Clients participating in the surgery
4. UserInControl: the User.cs object that represents the user that is currently in control of the surgical robot.

Functions

1. Defines a structure for a network of TelSurge users all participating in a single surgery
2. Allows updating with a received Surgery.cs object

User.cs

This class contains all the details of a TelSurge user including a user's:

- Name
- IP address
- if the user has omni controllers
- Omni controller positions
- "Frozen" status
- is a Master or a Client
- is in control of the surgical robot

- is "following" the movements of another surgeon's controllers
- haptic forces being sent to omni controllers
- the last time the users reported to the Master

Functions

1. Defines a TelSurge User
2. Gets current Omni controller positions
3. Sets haptic forces on Omni controllers
4. Allows Omnis to follow other Omnis
5. Sets a user's IP address to use with TelSurge
6. Initializes Omni controllers
7. Connects to Master
8. Checks emergency switch status
9. Checks "following" status
10. Checks "frozen" status
11. Checks an external button's status
12. Connects to external buttons (foot pedals)
13. Disconnects from Omni controllers

VideoCapture.cs

This class handles capturing video from a selected video source and displays it on the video viewer in TelSurgeMain.cs. This is ran as a separate thread and spawned from TelSurgeMain.cs.

Functions

1. Defines a class to handle video capturing from a device
2. Listens for video from the Master
3. Processes each video frame that comes in
4. Adds any telestration to each video frame
5. Compresses each video frame to JPEG with a specified amount of loss
6. Sends video frames to all clients (if Master)