

People Tracking With the Gantry-PTU 5-DOF Robot

07/19/2000

1. Introduction:

The 5-DOF ceiling-mounted gantry robot can track a person walking around the gantry's workspace, as shown in Figure 1. The demo is a seven-step procedure:

1. Position the camera using the gantry's teach pendant
2. Orient the camera with the PTU to point West (see Figure 1 for Cardinal points).
3. Execute **C:\POH\TRACKING\GMOVE2_3.EXE** on Robocop
4. Execute **~paul/gantry-demos/Hybrid5_1** on Scallop
5. Enter gains such as 1.0 for the PTU and 2.0 for the gantry
6. Place SSD over target of interest. Tracking will now follow the person
7. Pushing the gantry's pendant red abort button will terminate tracking. Kill the **Hybrid5_1** Scallop process with **CTRL-C** and kill the Robocop **GMOVE2_3** program with **ESC** and halt the PTU.

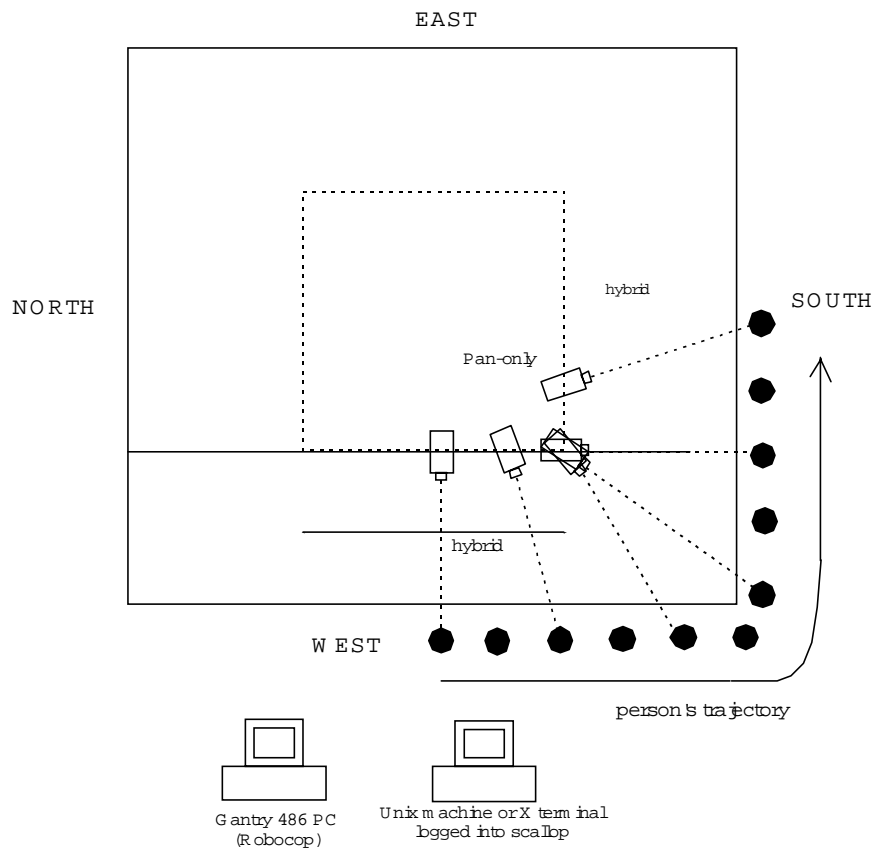


Figure 1: Top view of gantry workspace. Room's south side points to Uris Hall

2. Details of the Seven Steps

2.1 Step One: Gantry camera positioning

The gantry is typically always left on and its teach pendant effects gantry motions. Gantry positioning places the target in the camera's general field-of-view. If this is so, then one can proceed to Step Two. Occasional power surges or Robocop hard reboots however, can cause the gantry to freeze and be unresponsive to teach pendant commands. If this is so, the gantry must be rebooted and re-homed.

2.1.1 Rebooting:

Rebooting the gantry involves first, powering it down then up and second, reloading the gantry's AT6400 operating system. Power down is achieved by switching off the yellow power bar. Since, the gantry will immediately translate downwards, it is important to gently support the gantry end-effector while powering down. Switching power on again, one will hear the gantry lock into position and one can now freely remove support.

Executing **AT6400.EXE** on Robocop will load the gantry's operating system. Once loaded, the gantry must be initialized via the **FASTTERM.EXE**, terminal program:

C:\POH\SETUP\FASTTERM.EXE

Accept the 768 default base port address and load the gantry initialization program:

Hit F1

Type: **C:\POH\SETUP\INIT.PRG <ENTER>**

<ENTER>

Type: **psetup <ENTER>**

The gantry has been rebooted and initialized. The gantry's teach pendant will XYZ jogging after one types:

ls 333 <ENTER>

lh 333 <ENTER>

jog 111 <ENTER>

2.1.2 Re-homing the Gantry

The gantry's primary home position is in the North-West corner (see Figure 1). First, bring the gantry close to this corner with the teach pendant, then **FASTTERM** type:

hom010 <ENTER>

The gantry should begin slowly homing. For people tracking, a secondary home position was defined and lies appropriately halfway between the North and South sides of the gantry as shown in Figure 1. Under **FASTTERM**, one can load and execute the secondary home program:

Hit **F1**

Type: **C:\POH\TRACKING\HOMEME2.PRG <ENTER>**

Type: **homme2 <ENTER>**

The gantry can be further positioned if desired with the teach pendant. This may be necessary to get a good camera field-of-view over the target of interest; one can jog the camera vertically, horizontally and set a camera-to-target depth.

2.2 Step Two: PTU Orientation

The PTU is serially tethered to Scallop via the Central Data Quad-Serial Port Box on port **/dev/sts/ttyC53** (case sensitive). Via **KERMIT**, one can execute PTU commands:

```
bash$ kermit -l \dev\sts\ttyC53
```

Type: **c**

Type: **pp-1750**

Type: **tp0**

Type: **ee**

Type: **ft**

Pan position, **pp-1750** and tilt position, **tp0**, orients the camera's optical axis West as shown in Figure 1. **KERMIT** must be exited (using the key combination **<CTRL>-\
<CTRL>-C**) before tracking begins. At anytime, PTU motions can be halted by typing **H** under **KERMIT**, or by switching off the PTU power supply.

2.3 Step 3: Executing *GMOVE2_3.EXE*

GMOVE2_3 in **C:\POH\TRACKING** is a DOS program on Robocop. It executes gantry motion requests issued by Scallop. **GMOVE2_3** must be running before Step 4. It's source code is in the same directory. At essence, **GMOVE2_3** receives and transmits ASCII characters, which the AT6400 recognizes as gantry motion commands.

2.4 Step 4: Executing *Hybrid5_1*

Hybrid5_1 in **~paul/gantry-demos** is a GCC compiled executable and must be run while logged into Scallop. **Hybrid5_1** begins checking the serial connection between Scallop and Robocop (**/dev/sts/ttyb**) and the serial connection between Scallop and the PTU (**/dev/sts/ttyC53**) and hence **KERMIT** must not be running simultaneously.

Hybrid5_1 calls X-Vision functions which in turn call X11 routines. The current `~paul/.profile LD_LIBRARY_PATH` may be incorrect and **Hybrid5_1** will complain. Unsetting this path will overcome this problem.

2.5 Step 5: Setting Gains

Executing **Hybrid5_1** will prompt the user to enter gains. This program suggests a PTU gain of 1.0 (a float) and a gantry gain of 2.0 (a float). Larger PTU gains ($1.0 < \text{PTU gain} < 2.0$) and a gantry gain of 2.0 will track a faster moving person. PTU gains > 2.0 can yield unstable results and should be avoided.

2.6 Step 6: SSD Placement

After gains entry, **Hybrid5_1** will display what the camera sees and an 80x80 SSD tracking window. Place the SSD window with the mouse, over the person's head and click. Partitioned tracking will begin and the person can freely walk around the gantry workspace.

The camera view and SSD window are displayed on the black and white video monitor. If the SSD loses the target, it is important to quickly kill the gantry (teach pendant abort push button), kill the **Hybrid5_1** process, kill **GMOVE2_3** and halt the PTU. This same procedure is used to terminate the demo and is explained in the following steps.

2.7 Step 7: Terminating the Demo

Pushing the red button on the gantry teach pendant at anytime, will abort all gantry motions. This push button is the one located closest to the pendant's tether cable (see Figure 2)

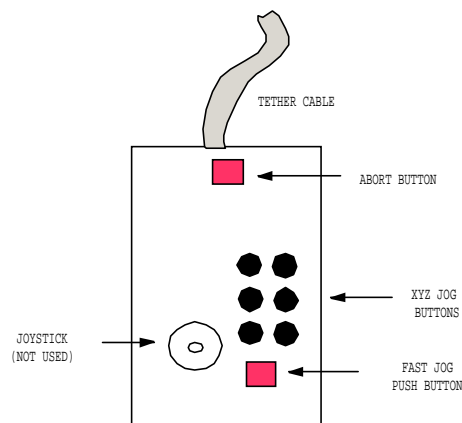


Figure 2: Gantry teach pendant

Pushing the abort button however will not terminate the **Hybrid5_1** process, nor the **GMOVE2_3** program. A simple **CTRL-C** will kill Scallop's **Hybrid5_1** process and

hitting **ESC** on Robocop will kill **GMOVE2_3**. The PTU should also be halted since it is possible that the PTU is still panning and/or tilting because it is executing the last command before **Hybrid5_1** was terminated. Under **KERMIT**, via **-l /dev/sts/ttyC53**, one types **H** and **<ENTER>** to immediately halt all PTU motions.

3. Appendix

This robot is a hybrid consisting of a variety of components, see Figure 3, which work together to track a moving target. Tracking will failure if any component is not properly configured or properly wired. This Appendix gives some common trouble-shooting tips and wiring diagrams and concludes with other directories of interest in **~paul**. This can be of help to future developers.

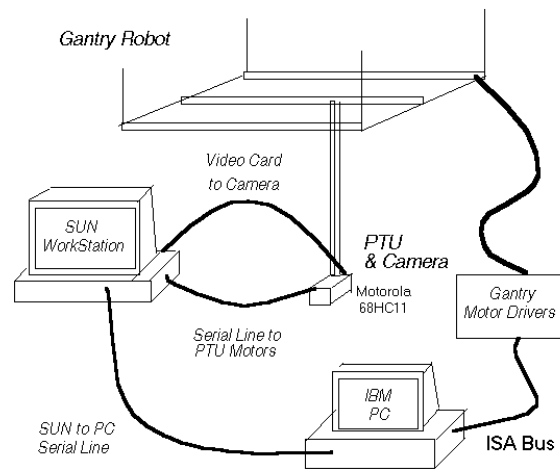


Figure 3: 5-DOF hybrid robot

The components are:

- Gantry robot and its 486 PC (named Robocop)
- Pan-tilt-unit (PTU) serially connect to Scallop off **/dev/sts/ttyb**
- K2T framegrabber installed in Scallop
- Scallop-Robocop serial connection off **/dev/sts/ttyC53**

3.1 The black and white monitor is not displaying what the camera sees

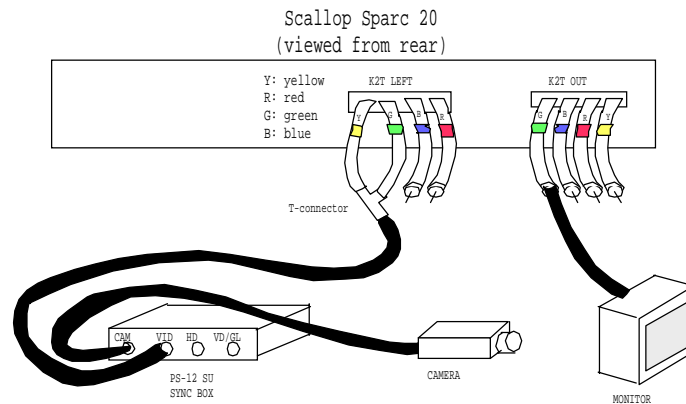


Figure 4: Camera, K2T framegrabber, PS-12 SU sync box and monitor cabling

Figure 4 illustrates the camera/K2T/sync-box/monitor wiring. The monitor should display what the camera sees if one uses the cabling in Figure 4. Refreshing the K2T framegrabber can also help using the `go_live` program:

`~paul/gantry-vision/k2t/examples/go_live`

Note the colored cables and their connections depicted in Figure 4. **Hybrid5_1** is hard-coded to use camera images through the K2T green channel.

3.2 Hybrid5_1 complains about serial connections

There are two serial connections in the setup. The first is the serial connection from Scallop's `/dev/sts/ttyb` serial port to Robocop's **COM1** serial port. If serial cable between these two computers is not connected, **Hybrid5_1** will complain and state a failure to establish gantry communications. It is thus important to check that this serial cable is properly connected.

The second serial connection is between Scallop's `/dev/sts/ttyC53` and the PTU. The Central Data quad serial port box provides the `ttyC53` serial port. **Hybrid5_1** is hard-coded to work with `/dev/sts/ttyC53` (case-sensitive). **Hybrid5_1** will complain of a failure to establish PTU communications if this port is not working. If one can use **KERMIT** to issue PTU ASCII commands with this port, then **Hybrid5_1** should also work with this port. Again, it is important to make sure this serial cable is properly connected. For reference, Figure 5 is the serial cable pin out between Scallop and the PTU.

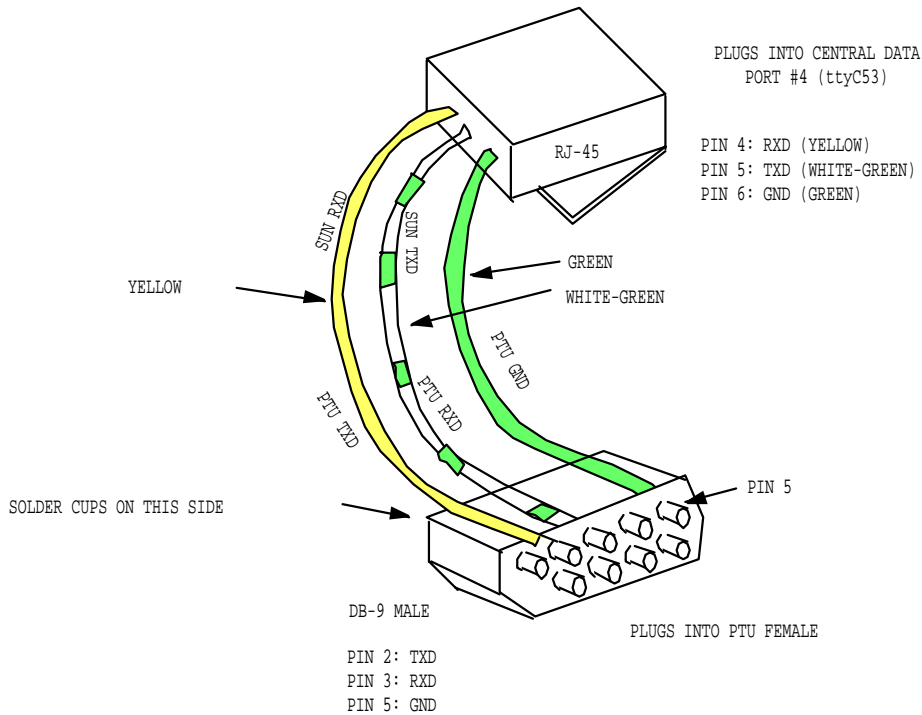


Figure 5: PTU-to-Scallop serial cable pin out

3.3 KERMIT works with the PTU, but Hybrid5_1 still complains

Hybrid5_1 is hard-coded to work with the PTU under terse feedback terse (**ft**) and echo enabled (**ee**). Section 2.2 gave details on setting this PTU configuration:

```
bash$ kermit -I \dev\sts\ttyC53
Type: c
Type: pp-1750
Type: tp0
Type: ee
Type: ft
```

Hybrid5_1 reads strings generated by the PTU. Terse feedback returns shorter ASCII strings than under verbose feedback. Return strings commence with an asterisk which is **Hybrid5_1** uses. Some other executables in **~paul/Gantry-partitioning** turn echoing off to decrease program cycle time (thus increasing bandwidth), but **Hybrid5_1** in the demo uses echoing.

3.4 pp-1750 doesn't orient the PTU west

pp0 points the camera south (towards Uris Hall) and is the PTU's default home orientation. Issuing -1750 steps clockwise (as viewed from above) via a **pp-1750**

command should point the camera west. The PTU resolution is 0.0514 degrees/step, thus $0.0514 * 1750 \text{ steps} \sim 90 \text{ degrees}$.

If **pp0** doesn't point the camera south, then possible the PTU was mounted on the gantry end effector in the reverse orientation. The camera would then be pointing north. Because **Hybrid5_1** couples the PTU orientation to the gantry's X-axis (Motor 1), the PTU should be re-mounted to point the camera south when at **pp0**.

By default, the PTU is configured with limits enabled (ASCII command **le**). If not enabled, a pan orientation beyond ± 3000 steps or a tilt orientation beyond ± 600 steps could mechanically force the PTU past its Hall effect limit switches. When this occurs, the PTU will no longer be calibrated properly; issuing **pp0** will fail to servo the PTU to a right-angle orientation. Under these circumstances, the PTU typically grinds (having hit its hard limit) after issuing a reset command (ASCII command **r**). To overcome this, one has to open up and manually turn the PTU into its default PTU orientation. The PTU housing can be opened with an Allen key, and oriented by turning the gears by hand. A **r** command should then reset the PTU properly.

3.5 Other source code and executables in ~paul

Directories of interest to developers under **~paul** are:

gantry-serial:	examples in Scallop-to-Robocop serial communications
gantry-demos:	various demos
gantry-partition:	partitioned control development
gantry-depth:	recursive least squares camera-to-target depth development
gantry-programs:	array handling functions
gantry-test:	AT6400 .prg code - loaded and executed under Robocop
gantry-dperception:	Directed Perceptions PTU binaries (compiled by Atanas)
gantry-time:	Unix gettimeofday() examples for timing program cycle time
gantry-figures:	figures used in ICRA publications and thesis
gantry-ptu:	programs to servo the PTU with serial ASCII commands
gantry-tracking:	3 DOF pose regulation (block tracking with 4 fiducials)
gantry-ptu-tracking:	5 DOF pose regulation development
gantry-vision:	K2T code, old and new versions of X-Vision
gantry-kalman:	examples of Kalman filter for estimating camera-to-target depth
gantry-rccl:	Vicky Puma 560 servo programs using RCCL libraries
Gantry-binarized:	partitioning using Directed Perceptions' PTU binaries
Gantry-sin:	various programs used for Bode system ID of gantry and PTU
Gantry-time:	programs used to measure program cycle time
Gantry-depth:	recursive least squares for measuring camera-to-target depth
Gantry-partition:	people tracking, corner handling, depth handling development

Each directory has a **00index** text file which describes directory contents. There is a distinction between directory names with the leading gantry- and Gantry- prefix. The former are programs compiled with the an older version (prior to January 1999) of X-Vision. The latter are programs compiled with the newest version of X-Vision (as of July 2000). This X-Vision release has additional features like SSD scale factors (used in depth regulation programs) and frame-acquisition timing. I used the convention of leading a filename with an uppercase if it was compiled using the newer X-Vision version (e.g. **Hybrid5_1**). Lowercase was used if compiled in the older version (e.g. **hybrid5_1**).

Robocop also has useful code under the **C:\POH** directory.

TRACKING:	GMOVE development (compiled under Borland C DOS version)
SETUP:	AT6400 gantry initialization setup programs
MYPROGS:	AT6400 programs (*.prg) written for commanding gantry motions

Paul Oh
July 2000