Absolutely. Based on everything we just worked through, here is a **complete, reproducible troubleshooting guide** for getting the **HTC Vive Pro 2 working on Fedora/Linux with the CertainLach SteamVR driver**, including the specific **lens-server / Wine / Proton problem** that caused the crashing and upside-down/distorted image.

The important part is that your final working setup **uses Steam's Proton Experimental for `lens-server.exe`**, even though SteamVR itself exposes `WINE=/run/host/usr/bin/wine` and does not provide the normal Proton compatibility variables.

The upstream driver does support using Steam's Proton, and the repository's current code contains the Proton fallback mechanism. (GitHub)

HTC Vive Pro 2 on Fedora Linux — Complete SteamVR Fix Guide

What this fixes

This guide is aimed at problems such as:

- SteamVR starts but the Vive Pro 2 image is upside down.
- SteamVR crashes when the HMD initializes.
- `failed to wrap hmd`
- `lens client error`
- `can't find wine in PATH`
- `io error: No such file or directory`
- `Lens distortion helper is failed to launch`
- `lens-server.exe` doesn't start.
- SteamVR cannot find/run the Vive Pro 2 distortion helper.
- Fedora's SteamVR environment provides `/run/host/usr/bin/wine`, but that path doesn't exist.
- Building the driver with stable Rust fails because of `#![feature(try_blocks)]`.

The official driver works by intercepting SteamVR's normal Lighthouse driver and providing Vive Pro 2-specific functionality. The Windows-only Vive lens distortion library is run through `lens-server.exe`. (GitHub)

---

1. Install SteamVR

Install SteamVR normally through Steam.

The expected location in this guide is:

```
~/.local/share/Steam/steamapps/common/SteamVR
```

Check:

```
ls -ld "$HOME/.local/share/Steam/steamapps/common/SteamVR"
```

You should see the directory.

If SteamVR is installed somewhere else, the driver installer supports overriding the location with `STEAMVR`.

For example:

```
export STEAMVR="$HOME/.local/share/Steam/steamapps/common/SteamVR"
```

---

2. Get the Vive Pro 2 Linux driver

Clone the driver:

```
cd ~
git clone https://github.com/CertainLach/VivePro2-Linux-Driver.git
cd ~/VivePro2-Linux-Driver
```

Or, if you already have it:

```
cd ~/VivePro2-Linux-Driver
```

The upstream project is here:

CertainLach/VivePro2-Linux-Driver

The project itself recommends either a Nix build, manual building, or the packaged release, followed by its installer. (GitHub)

---

3. Make sure the lens-server files exist

This is **very important**.

Check:

```
ls -lah \
"$HOME/.local/share/Steam/steamapps/common/SteamVR/drivers/lighthouse/bin/
linux64/lens-server/"
```

You need:

```
lens-server.exe
LibLensDistortion.dll
opencv_world346.dll
```

For example:

```
lens-server.exe
LibLensDistortion.dll
opencv_world346.dll
```

The upstream project specifically expects `lens-server.exe` and the two DLLs in this directory. (GitHub)

If they aren't there, the lens-server portion of the driver cannot work.

---

4. Check Wine

The original driver can use Wine.

Check:

```
command -v wine
command -v wine64
```

For example, you might get:

```
/usr/bin/wine
/usr/bin/wine64
```

Test it:

```
wine --version
wine64 --version
```

In our case, Wine itself worked.

We also manually tested the lens server:

```
WINE=/usr/bin/wine64 \
"$HOME/.local/share/Steam/steamapps/common/SteamVR/drivers/lighthouse/bin/
linux64/lens-server/lens-server.exe"
```

And it successfully reported:

```
INFO lens_server: hello from lens server
INFO lens_server: dll path: ".../LibLensDistortion.dll"
```

That proves:

Wine + lens-server.exe + LibLensDistortion.dll + OpenCV were functional.

---

5. The important SteamVR problem

This is where the problem gets interesting.

SteamVR was launching the driver with:

```
WINE=/run/host/usr/bin/wine
```

but inside the SteamVR environment:

```
/run/host/usr/bin/wine
```

didn't exist.

We confirmed this from the SteamVR process:

```
tr '\0' '\n' < /proc/$(pgrep -n vrserver)/environ | \
grep -E '^(STEAM_COMPAT|PROTON|WINE|PATH)='
```

The result was:

```
PATH=/usr/bin:/bin
WINE=/run/host/usr/bin/wine
```

But:

```
ls -l /run/host/usr/bin/wine
```

would show that it isn't available from the SteamVR environment.

This resulted in the driver failing to use the Wine path supplied by SteamVR.

---

6. Why Proton is the better solution

Steam already contains Proton.

Check:

```
ls -lah \
"$HOME/.local/share/Steam/steamapps/common/Proton - Experimental/"
```

You should see:

```
proton
files/
version
...
```

The important executable is:

```
~/.local/share/Steam/steamapps/common/Proton - Experimental/proton
```

The driver already contains code to use Proton as a fallback.

The problem in our case was that SteamVR **did not provide**:

```
STEAM_COMPAT_INSTALL_PATH
STEAM_COMPAT_DATA_PATH
PROTON_VERSION
```

We checked:

```
echo "STEAM_COMPAT_INSTALL_PATH=$STEAM_COMPAT_INSTALL_PATH"
echo "STEAM_COMPAT_DATA_PATH=$STEAM_COMPAT_DATA_PATH"
echo "PROTON_VERSION=$PROTON_VERSION"
```

and got:

```
STEAM_COMPAT_INSTALL_PATH=
STEAM_COMPAT_DATA_PATH=
PROTON_VERSION=
```

Therefore the original automatic Proton-selection logic couldn't discover Proton through those variables.

---

7. Modify `lens-client`

Open:

```
cd ~/VivePro2-Linux-Driver
nano crates/lens-client/src/lib.rs
```

The important change is to make the driver:

1. Try `WINE`.
2. If `WINE` doesn't actually exist, don't fail.
3. Fall back to Steam's Proton.
4. Find Proton Experimental manually if SteamVR hasn't supplied Proton variables.
5. If Proton doesn't work, fall back to normal `wine64` / `wine`.

Our modified `start_lens_server()` did exactly that.

The key behavior is:

```
if let Some(wine) = env::var_os("WINE") {
    let wine_path = PathBuf::from(&wine);

    if wine_path.exists() {
        info!("using WINE from environment: {}", wine_path.display());
        ...
    }

    warn!(
        "WINE points to {}, but it does not exist; falling back to Proton/wine
search",
        wine_path.display()
    );
}
```

Then it searches for Proton.

And finally:

```
for wine in ["wine64", "wine"] {
    ...
}
```

This is important because SteamVR's `WINE` variable can exist while pointing at an inaccessible path.

---

8. Switch to Nightly Rust

The driver currently requires unstable Rust features.

When we initially tried:

```
cargo build --release
```

with stable Rust, we got:

```
error[E0554]: `#![feature]` may not be used on the stable release channel
```

specifically:

```
#![feature(try_blocks)]
```

Therefore use nightly.

Check your toolchains:

```
rustup toolchain list
```

You should have something similar to:

```
stable-x86_64-unknown-linux-gnu
nightly-x86_64-unknown-linux-gnu
```

Set the repository to nightly:

```
cd ~/VivePro2-Linux-Driver
rustup override set nightly
```

Then verify:

```
rustc --version
cargo --version
rustup show active-toolchain
```

You should see:

```
nightly-x86_64-unknown-linux-gnu
```

---

9. Build the driver

Now:

```
cd ~/VivePro2-Linux-Driver
cargo build --release
```

A successful build ends with:

```
Finished `release` profile [optimized] target(s)
```

Check:

```
ls -lh target/release/libdriver_proxy.so
```

You should have:

```
libdriver_proxy.so
```

Warnings aren't necessarily a problem.

For example, we got warnings about unused imports and unused features, but the build succeeded.

---

10. Replace the proxy binary

The packaged installer expects:

```
dist-proxy/driver_lighthouse.so
```

while Cargo creates:

```
target/release/libdriver_proxy.so
```

Copy the new build:

```
cd ~/VivePro2-Linux-Driver
```

```
cp -f \
target/release/libdriver_proxy.so \
dist-proxy/driver_lighthouse.so
```

Verify:

```
file dist-proxy/driver_lighthouse.so
file target/release/libdriver_proxy.so
```

Both should report:

```
ELF 64-bit LSB shared object, x86-64
```

And the Build IDs should be identical.

---

11. Verify that your modified code is actually inside the binary

This is a really useful diagnostic step.

Run:

```
strings dist-proxy/driver_lighthouse.so | grep -E \
'using WINE from environment|server is working|WINE points to|trying .* as
proton|missing proton prefix' \
| head -20
```

You should see strings such as:

```
using WINE from environment:
WINE points to
server is working
trying ... as proton
missing proton prefix environment variable
```

If you don't see those strings, you probably copied the wrong binary or built the wrong source.

---

12. Stop SteamVR completely

Before installing the new driver:

```
steamvr_shutdown 2>/dev/null || true

pkill -f vrserver || true
pkill -f vrcompositor || true
pkill -f vrmonitor || true

sleep 2
```

Verify:

```
pgrep -a 'vrserver|vrcompositor|vrmonitor' || \
echo "SteamVR processes stopped."
```

You want:

```
SteamVR processes stopped.
```

---

13. Install the modified driver

The repository does **not** have:

```
./install.sh
```

at its root.

The installer is:

```
dist-proxy/install.sh
```

So:

```
cd ~/VivePro2-Linux-Driver/dist-proxy
./install.sh
```

It should locate SteamVR automatically:

```
SteamVR at /home/lytra/.local/share/Steam/steamapps/common/SteamVR
```

Then:

```
= Overriding current driver
= Updating proxy server
Installation finished, try to start SteamVR
```

About this warning

You may see:

```
source pattern not found, file is already patched?
one or more rules failed
```

during:

```
= Patching real driver
```

That doesn't necessarily mean installation failed.

In our case the installer continued and successfully replaced:

```
driver_lighthouse.so
```

and copied the lens-server files.

---

14. Verify the installed driver

Run:

```
strings \
"$HOME/.local/share/Steam/steamapps/common/SteamVR/drivers/lighthouse/bin/
linux64/driver_lighthouse.so" \
| grep -E \
'using WINE from environment|server is working|WINE points to|trying .* as
proton|missing proton prefix' \
| head -20
```

You should see the modified strings.

That confirms that SteamVR is going to load the new binary.

---

15. Start SteamVR

Start SteamVR normally.

Then immediately check:

```
grep -nE \
'lens_client|lens_server|WINE|Proton|server is working|cant find server|fatal
error|distortion' \
"$HOME/.local/share/Steam/logs/vrserver.txt" \
| tail -100
```

The important successful sequence should look roughly like:

```
INFO lens_client: using lens server at ".../lens-server.exe"
```

```
WARN lens_client: WINE points to /run/host/usr/bin/wine, but it does not exist;
falling back to Proton/wine search
INFO lens_client: trying .../Proton - Experimental/proton as proton
```

The critical point is that it **doesn't crash afterward**.

---

16. Verify that Proton actually launched lens-server

Run:

```
pgrep -af 'lens-server|wine|proton'
```

In our working setup we got:

```
.../Proton - Experimental/proton runinprefix start.exe /wait /i /b /unix
.../lens-server.exe
```

and:

```
.../Proton - Experimental/files/.../wineserver
```

and:

```
C:\windows\system32\start.exe ...
```

and finally:

```
.../lens-server.exe
```

This proves the chain is:

```
SteamVR
   │
   ▼
Vive Pro 2 proxy driver
   │
   ▼
lens-client
   │
   ▼
Steam Proton Experimental
   │
   ▼
Wine
   │
   ▼
lens-server.exe
   │
   ├── LibLensDistortion.dll
   │
   └── opencv_world346.dll
```

---

17. Verify lens-server isn't crashing

Check:

```
pgrep -af 'lens-server.exe|start.exe.*lens-server|Proton - Experimental/proton'
```

You should see the Proton process and lens-server process while SteamVR is running.

You can also check:

```
ps -ef | grep '[l]ens-server'
```

If it is working, you shouldn't see SteamVR repeatedly restarting it.

---

18. Check the SteamVR log for distortion processing

This is another excellent test:

```
grep -nE \
'server is working|lens protocol|LensClient|lens_client|fatal error|failed to
launch|distortion|ComputeDistortion' \
"$HOME/.local/share/Steam/logs/vrserver.txt" \
| tail -100
```

A good sign is:

```
Processing message VRMsg_ComputeDistortionBatched
```

For example, we saw:

```
Processing message VRMsg_ComputeDistortionBatched ...
```

multiple times.

That means SteamVR is actively requesting distortion calculations.

---

19. Check that SteamVR itself is stable

Run:

```
pgrep -a 'vrserver|vrcompositor|vrmonitor'
```

You should see:

```
vrserver
vrcompositor
vrmonitor
```

If those remain running and the HMD image is visible, the basic driver chain is functioning.

---

20. The most important diagnostic commands

If somebody else has this problem, these are the commands I'd have them run first.

Check driver

```
ls -lh \
"$HOME/.local/share/Steam/steamapps/common/SteamVR/drivers/lighthouse/bin/
linux64/driver_lighthouse.so"
```

Check lens server

```
ls -lh \
"$HOME/.local/share/Steam/steamapps/common/SteamVR/drivers/lighthouse/bin/
linux64/lens-server/"
```

Expected:

```
lens-server.exe
LibLensDistortion.dll
opencv_world346.dll
```

Check Wine

```
command -v wine
command -v wine64
wine --version
```

Check Proton

```
ls -lh \
"$HOME/.local/share/Steam/steamapps/common/Proton - Experimental/proton"
```

Check SteamVR's environment

```
tr '\0' '\n' < /proc/$(pgrep -n vrserver)/environ | \
grep -E '^(STEAM_COMPAT|PROTON|WINE|PATH)='
```

Check lens-server

```
pgrep -af 'lens-server|wine|proton'
```

Check driver errors

```
grep -nE \
'lens_client|lens_server|WINE|Proton|fatal error|failed to launch|distortion' \
"$HOME/.local/share/Steam/logs/vrserver.txt" \
| tail -100
```

---

21. What the important errors mean

`can't find wine in PATH`

The driver couldn't find either:

```
wine64
```

or:

```
wine
```

in the SteamVR environment.

The upstream project historically expected Wine for the Windows-only lens library. (GitHub)

Our modification adds a Proton fallback.

`WINE=/run/host/usr/bin/wine`

This isn't necessarily useful.

The important question is:

```
test -x /run/host/usr/bin/wine && echo yes || echo no
```

If it says:

```
no
```

then SteamVR's `WINE` variable points somewhere inaccessible/nonexistent.

Our modified driver detects this instead of blindly trying it.

`Proton: No compat data path?`

This happens if you manually run:

```
Proton - Experimental/proton runinprefix ...
```

without supplying a Proton compatibility-data environment.

So don't use that as the normal diagnostic.

The driver needs to establish/use its own prefix appropriately.

`source pattern not found, file is already patched?`

This comes from the `sewer` patching step.

If the SteamVR Lighthouse driver has already been patched, seeing this during a subsequent installation isn't automatically fatal.

The important part is whether:

```
= Overriding current driver
```

and:

```
= Updating proxy server
```

complete successfully.

Stable Rust:

```
error[E0554]: #![feature] may not be used on the stable release channel
```

Use:

```
rustup override set nightly
```

Then:

```
cargo build --release
```

The repository contains code requiring nightly features, so stable Rust can fail exactly as we encountered.

---

22. Don't accidentally use the old binary

This is an easy mistake.

Cargo produces:

```
target/release/libdriver_proxy.so
```

SteamVR needs:

```
driver_lighthouse.so
```

Therefore after every rebuild:

```
cp -f \
target/release/libdriver_proxy.so \
dist-proxy/driver_lighthouse.so
```

Then reinstall:

```
cd ~/VivePro2-Linux-Driver/dist-proxy
./install.sh
```

Then verify:

```
strings \
```

```
"$HOME/.local/share/Steam/steamapps/common/SteamVR/drivers/lighthouse/bin/
linux64/driver_lighthouse.so" \
| grep 'server is working'
```

---

23. Complete quick-fix recipe

Once someone understands the problem, the whole procedure can be condensed to:

```
cd ~/VivePro2-Linux-Driver

# Stop SteamVR
steamvr_shutdown 2>/dev/null || true
pkill -f vrserver || true
pkill -f vrcompositor || true
pkill -f vrmonitor || true
sleep 2

# Use nightly Rust
rustup override set nightly

# Build
cargo build --release

# Replace packaged proxy binary
cp -f target/release/libdriver_proxy.so \
      dist-proxy/driver_lighthouse.so

# Install
cd dist-proxy
./install.sh
```

Then start SteamVR and check:

```
grep -nE \
'lens_client|WINE|Proton|fatal error|failed to launch|ComputeDistortion' \
"$HOME/.local/share/Steam/logs/vrserver.txt" \
| tail -100
```

And:

```
pgrep -af 'lens-server|wine|proton'
```

---

24. What our successful result looked like

The final important evidence was:

```
WARN lens_client: WINE points to /run/host/usr/bin/wine, but it does not exist;
falling back to Proton/wine search
```

followed by:

```
INFO lens_client: trying /home/lytra/.local/share/Steam/steamapps/common/Proton
- Experimental/proton as proton
```

Then Proton actually launched:

```
Proton - Experimental/proton runinprefix start.exe /wait /i /b /unix .../lens-server.exe
```

and the lens server existed as a Wine process:

```
...lens-server.exe
```

SteamVR stayed alive:

```
vrserver
vrcompositor
vrmonitor
```

and SteamVR was processing:

```
VRMsg_ComputeDistortionBatched
```

Most importantly, **the headset was no longer upside down and SteamVR was no longer crashing**.

That's the point at which I'd consider the lens-server problem solved.

---

25. One important caveat

This fixes the **lens-server/Wine/Proton portion** of the Vive Pro 2 driver.

It does **not** guarantee that every Vive Pro 2 issue on Linux is solved. The upstream project still lists various limitations, and there are open reports concerning things such as visual alignment, double vision and newer SteamVR versions. (GitHub)

The current upstream README also documents several Vive Pro 2 resolution modes and notes that higher-resolution modes can require appropriate kernel/display support. (GitHub)

For a Fedora guide, I'd therefore describe this as:

> **"Fixing Vive Pro 2 SteamVR lens distortion / Proton startup on Linux"**

rather than claiming it is a universal fix for every VP2 problem.

Useful official references

VivePro2-Linux-Driver — CertainLach

HTC VIVE Pro 2 on Linux — santeri3700 guide

Vive Pro 2 driver issue tracker

The upstream project confirms that Steam's Proton support was specifically added as a way of eliminating the standalone Wine dependency, which is exactly the direction we ended up taking here. (GitHub)

And the key takeaway for anyone following this guide: if SteamVR says `WINE=/run/host/usr/bin/wine` but that path doesn't exist, **don't fight SteamVR's Wine environment**. Make the driver reject the invalid `WINE` path and fall back to **Steam's Proton Experimental**, then rebuild and reinstall the proxy. That was the change that got your setup from crashing/upside-down to stable. (GitHub)