

Remote Presence: Live Holograms for a Social Classroom

Tomi “bgt” Suovuo
Sebastian Hahta

3.2.2023



UNIVERSITY
OF TURKU

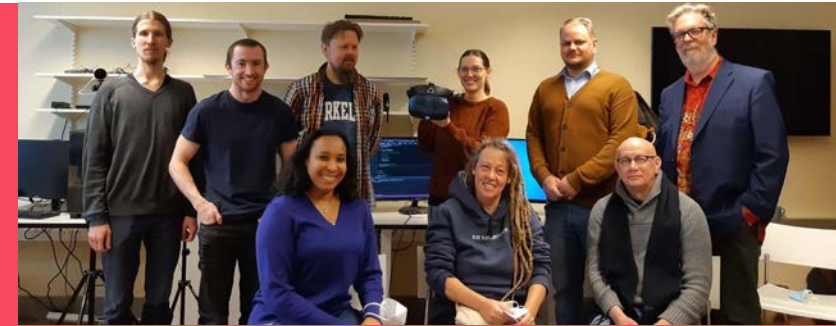
Beyond Video

Where might video and mixed reality be going next?

1. Our co-creation project
2. But why 3D video and holograms?
3. An overview of what we use
4. Demonstration...



Our Current Project BIT::TIP



- **Academy of Finland funded**
 - Co-create immersive video solutions
 - 6th graders and teachers
- Study the impact on social classroom climate and the sense-of-presence
 - Improve support for the social aspect of distance learning
- Evolve an end-to-end 3D capture and streaming platform
 - Image stability and completeness
 - Data compression and latency over the internet

<https://sites.utu.fi/bittip/>

Co-Creating



6th graders
and teachers

Holograms



- about 40ms latency excluding the network
- between 15 and 40 Mbit/s
- 30 frames per second
- Full HD resolution

Our Technology

Hardware



Intel RealSense L515
Depth Camera



Microsoft HoloLens 2
Headset



Dell XPS (GeForce RTX 2060)

Our Technology

Software

- Custom Capture, Rendering and Network
- Unreal Engine based application for HoloLens display
- Many HoloLens and Mixed Reality features provided by Unreal Engine (touch interaction, 3D sound, user interfaces)

```
continue;
}

LOG(INFO) << "RealSense intrinsic (" << profile.stream_name() << ") -
  << "fx: " << intrin.fx << ", fy: " << intrin.fy << ", "
  << "cx: " << -intrin.ppx << ", cy: " << -intrin.ppy << ", "
  << "w: " << intrin.width << ", h: " << intrin.height << ", "
  << "fps: " << profile.fps();

LOG(INFO) << "RealSense device: " << name_ << " #" << serial_;

rs2::pipeline_profile profile = pipe.start(cfg, [this](const rs2::frame &
try {
  // accelerometer rate is higher than color/depth, store in accel_data
  // without locking. Copied to frame when color and depth are received
  auto motion = f.as<rs2::motion_frame>();

  if (motion && motion.get_profile().stream_type() == RS2_STREAM_ACCEL) {
    motion.get_profile().format() == RS2_FORMAT_MOTION_XYZ32F) {

      rs2_vector accel_data_new = motion.get_motion_data();

      // accelerometer warning

      Eigen::Vector3f a_old(accel_data_.x, accel_data_.y, accel_data_.z);
      Eigen::Vector3f a_new(accel_data_new.x, accel_data_new.y, accel_data_new.z);
      float old_new_cos = a_old.dot(a_new)/(a_old.norm()*a_new.norm());

      if (abs((1.0f - old_new_cos) > accel_warning_threshold_) {
        // print warning (if enabled) and set cooldown counter
        if ((accel_warning_ctr_ == 0) && enable_accelerometer_warning_) {
          _sendWarning("Camera moved (" + name_ + ")? Accelerometer
            + "reading changed by "
            + std::to_string(acosf(old_new_cos) * (180.0f / M_PI))
            + " degrees from previous reading");
        }

        // always reset cooldown counter (cooldown count begins
        // only after accelerometer values have stabilized)
        accel_warning_ctr_ = accel_warning_timeout_;

      } else {
        accel_warning_ctr_ = std::max(accel_warning_ctr_ - 1, 0);
      }
    } // end of accelerometer warning

    accel_data_ = accel_data_new;

    // autocalibration

    // If camera has moved, start accumulating accelerometer values
    // at the end of the warning counter cooldown.
    if (autocalib_enable_ && (accel_warning_ctr_ == 1)) {
      _beginAutoCalibration();
    }
  }
}
```

Demonstration...



**UNIVERSITY
OF TURKU**

Questions?