

An Adaptive Resolution Scheme for Performance Enhancement of a Web-based Multi-User VR Application

Rishabh Pathak, Anderson Augusto Simiscuka and Gabriel-Miro Muntean

IEEE BMSB 2021

IEEE BMSB 2021

Online

Presenter: Anderson Augusto Simiscuka, Dublin City University

04/08/2021



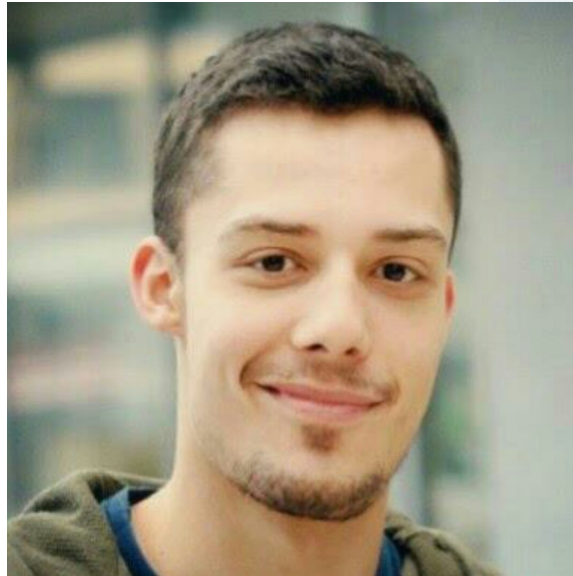
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 870610



Traction
Opera co-creation
for a social
transformation

Introduction

Anderson Augusto
Simiscuka



Postdoctoral Researcher

Doctor in Electronic Engineering

anderson.simiscuka2@mail.dcu.ie



Motivation

- The performance of web-based virtual reality (VR) applications, such as ones built with A-Frame, can be affected due to the limitations of the CPU and GPU, especially in multi-user applications.
- It is important to understand and analyze these applications' performance under different scenarios, demonstrating how an increase in the number of users affects metrics related to VR quality of experience (QoE).
- A smooth frame rate is important in VR applications to prevent motion sickness.



Multi-user VR Application

- We propose a **multi-user** VR application developed using the A-Frame library, rendered as a 360° experience within a browser.
- The application allows users to watch a film in a synchronized manner, in a **virtual cinema room**.



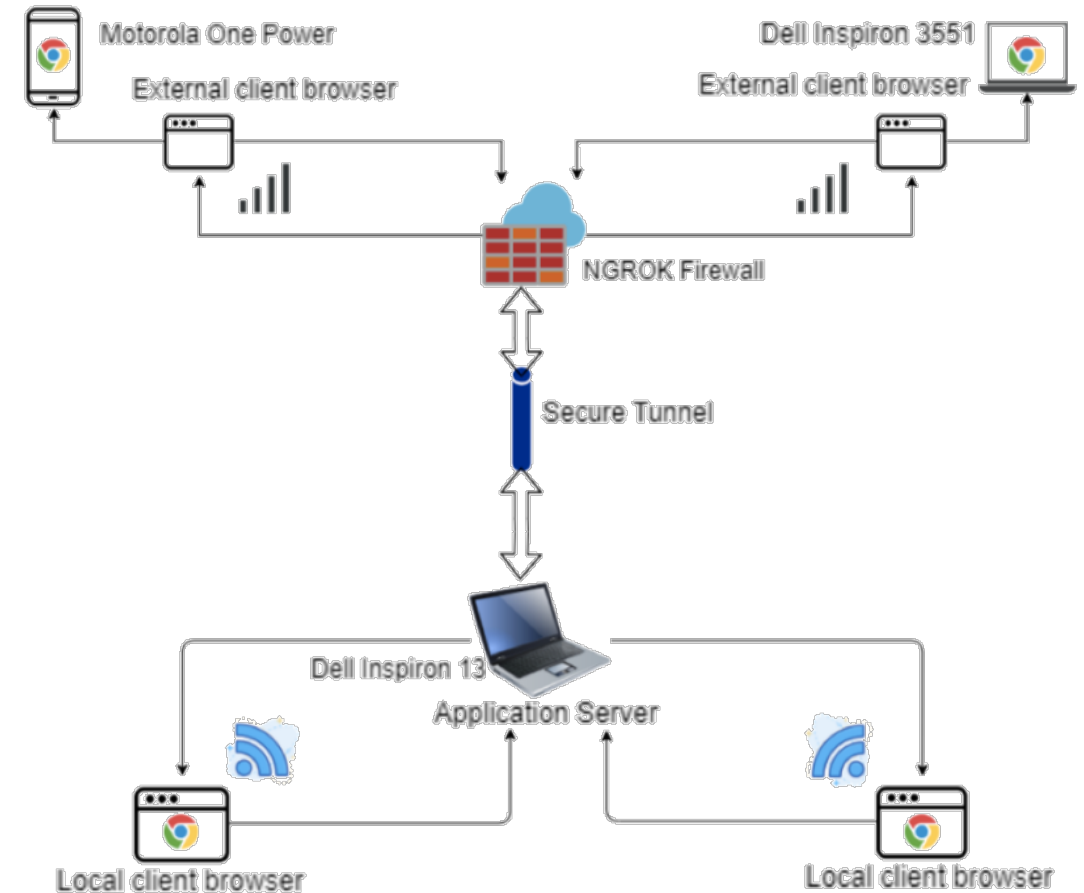
Multi-user VR Application

- The application employs a novel **Adaptive Resolution Scheme for VR (ARS-VR)**, which improves VR applications' frame rate and frame latency on remote devices with limited processing and display.
- Users can move in the application, while WebRTC allows for voice chats. The library Networked A-Frame broadcasts synchronous playback to all clients.



VR Application Architecture

- The application uses a reverse proxy service called **ngrok** to expose the local application server to a firewall on the public internet using secure tunnels.
- Users that are not on the same network as the server can access the application by querying the link to the **secure tunnel**.
- Two laptops were used as the application **server** and a **client**. A smartphone was used to simulate another client from a different network than the server.



Adaptive Resolution Scheme for VR

- Once a connection is established, the server starts sending application data to the client and **ARS-VR** starts.
- The algorithm is deployed with **JavaScript** and it adjusts the resolution of the application by monitoring the frame rate.
- The numerator and denominator of the aspect ratio are multiplied with the frame rate to provide a **dynamic resolution**.

Algorithm 1: Adaptive Resolution Scheme for VR

```
while browser_window==open do
  Input: framerate; screen_width;
        device_pixel_ratio; window_height
  Output: maxCanvasSize
  Function greatest_common_divisor(w,
    h):
    var w = screen_width*device_pixel_ratio
    var h = window_height*device_pixel_ratio
    var ratio = screen_width/screen_height
    var gcd = greatest_common_divisor(w, h)
    var num = w/gcd
    var den = h/gcd
    return num, den, ratio, w, h
  Function Main:
    if ratio < 1 then
      maxCanvasSize = {height: h/2, width: w/2}
    else
      maxCanvasSize = {height: framerate*den,
        width: framerate*num}
    return maxCanvasSize
end
end
```



Adaptive Resolution Scheme for VR

- This is the main **contribution** of our work, as the VR application was built in order to test the approach.
- The approach can be applied to other **web-based VR** experiences as well.
- With the browser rendering the 3D scenarios in a lower resolution, the framerate gets closer to **60fps**, which makes the VR experience more pleasant to users.

Algorithm 1: Adaptive Resolution Scheme for VR

```
while browser_window==open do
  Input: framerate; screen_width;
        device_pixel_ratio; window_height
  Output: maxCanvasSize
  Function greatest_common_divisor(w,
    h):
    var w = screen_width*device_pixel_ratio
    var h = window_height*device_pixel_ratio
    var ratio = screen_width/screen_height
    var gcd = greatest_common_divisor(w, h)
    var num = w/gcd
    var den = h/gcd
    return num, den, ratio, w, h
  Function Main:
    if ratio < 1 then
      maxCanvasSize = {height: h/2, width: w/2}
    else
      maxCanvasSize = {height: framerate*den,
        width: framerate*num}
    return maxCanvasSize
end
end
```



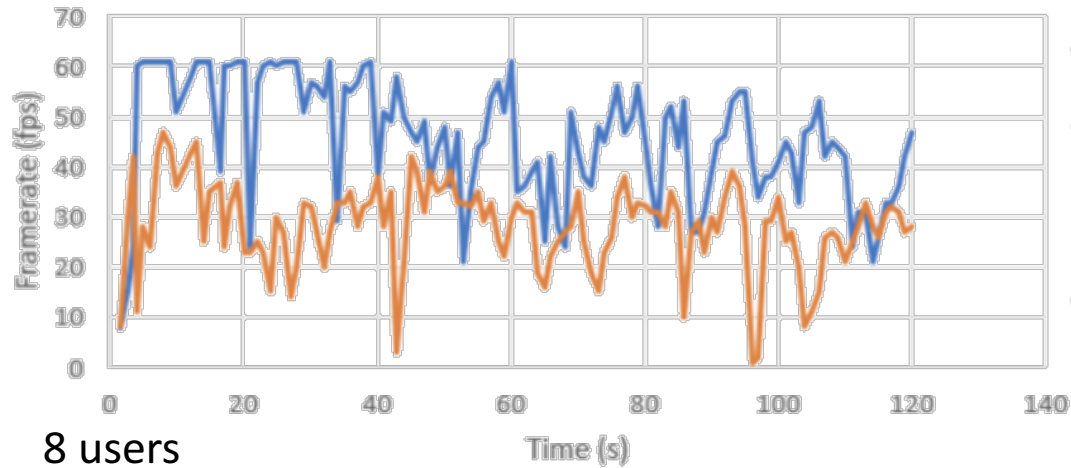
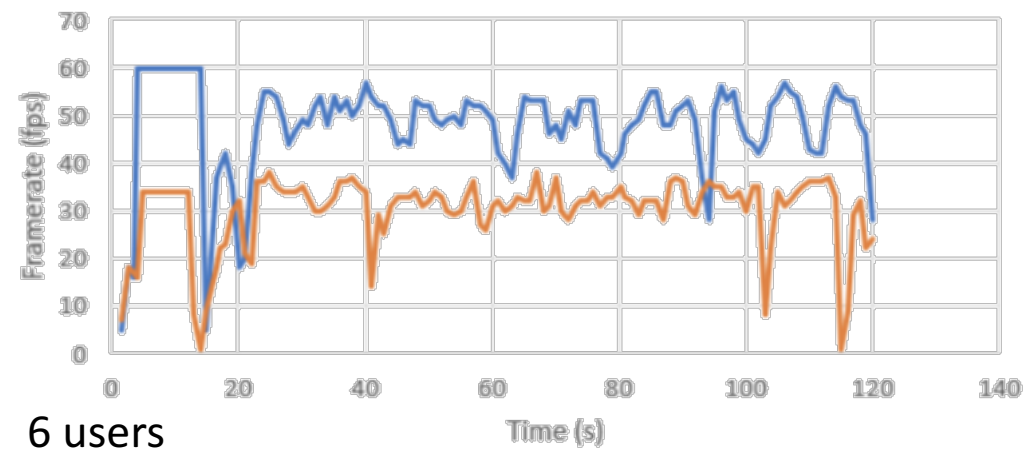
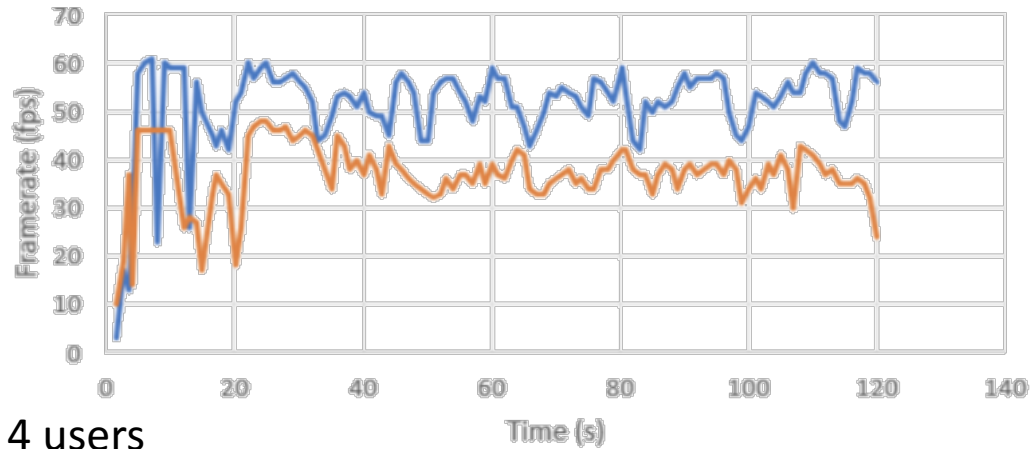
Performance Analysis

- Browser used was **Google Chrome**.
- The **frame rate** was collected for 120 seconds from the application with the algorithm 'off' (i.e. using the default 1920px resolution) and then with the algorithm 'on' (i.e. with dynamic resolution).
- Scenarios with **4, 6 and 8** simultaneous users were considered.



Performance Analysis - FPS

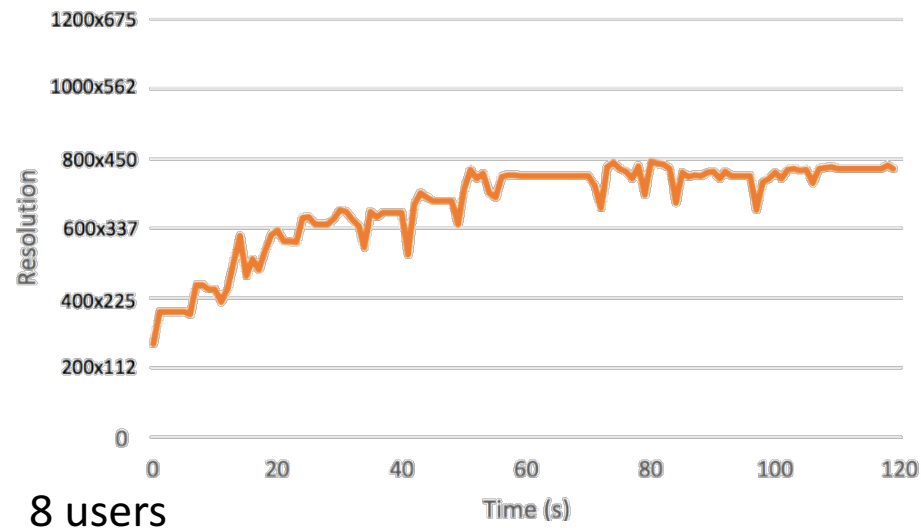
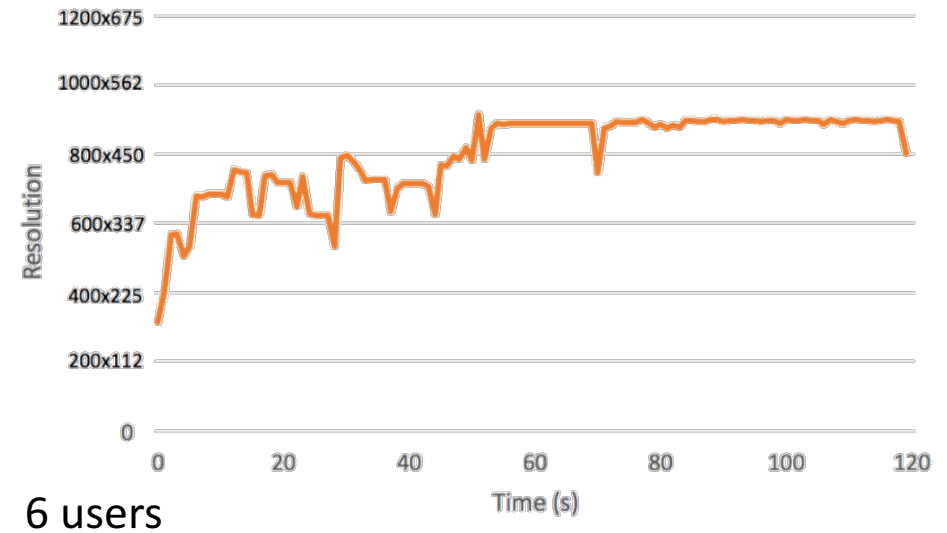
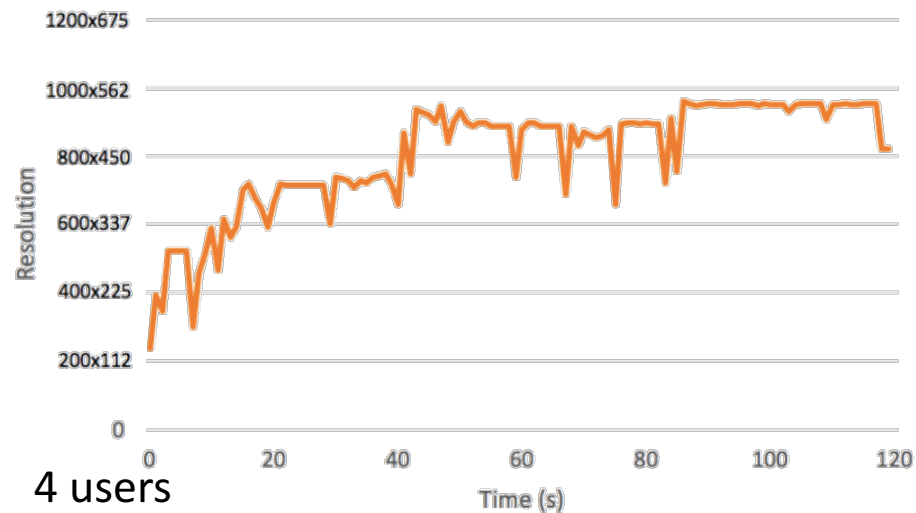
— Application runs with algorithm — Application runs without algorithm



No. users	Avg. fps	Avg. fps-ARS	Avg. resolution	FPS loss reduction
4	36.65	50.94	806x453	23%
6	29.69	47.33	801x450	30%
8	27.94	45.23	667x375	28%



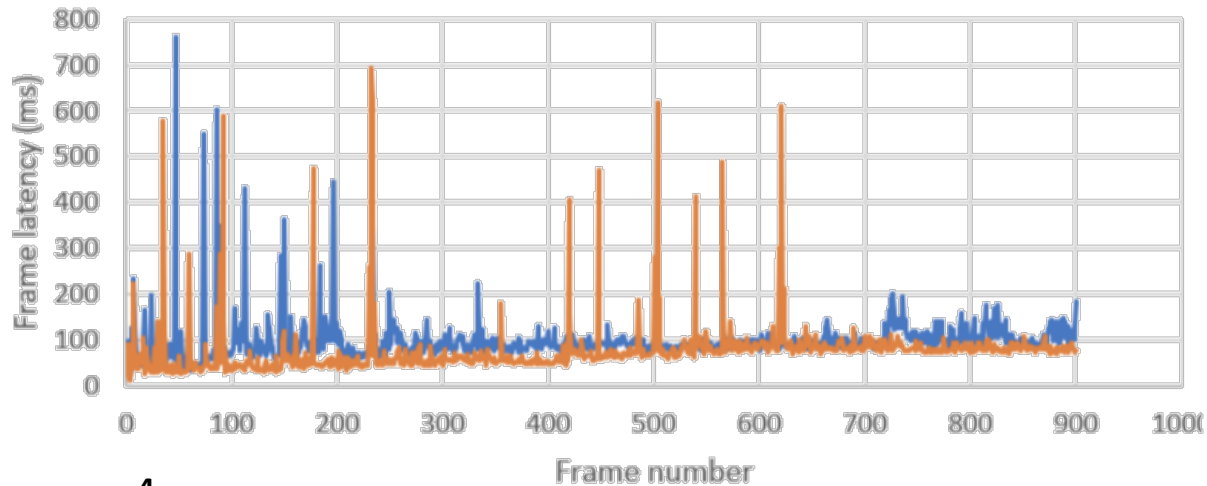
Performance Analysis - Resolution



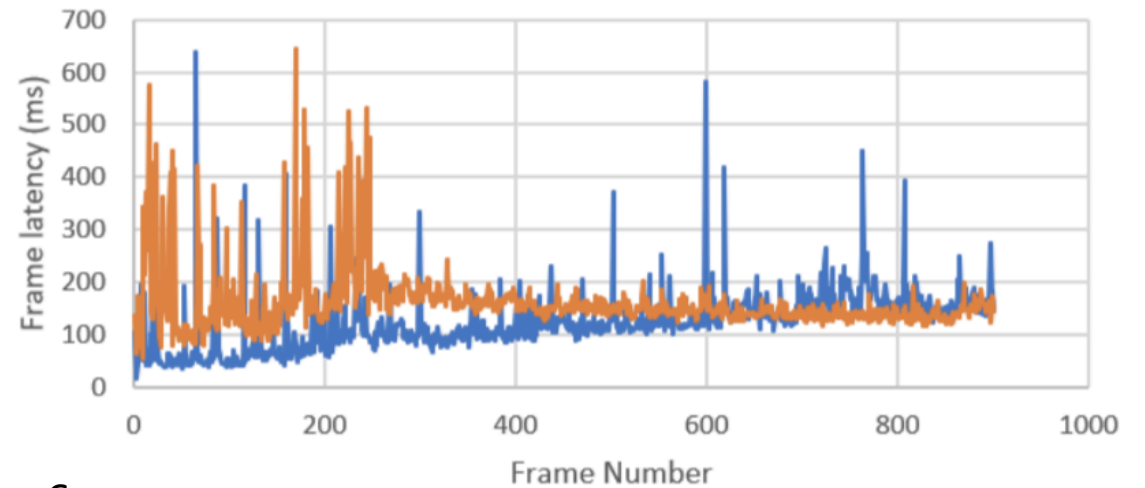
The resolutions, are on average 806x453px, 801x450px and 667x375px for the scenarios with 4, 6 and 8 users, respectively.



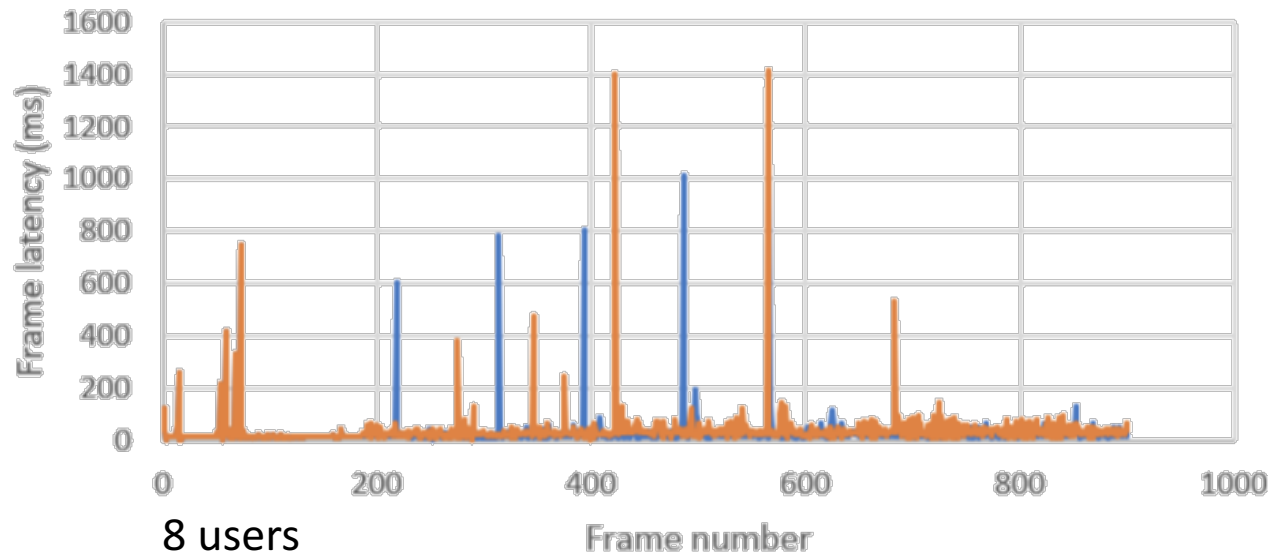
Performance Analysis - Frame Latency



4 users



6 users



8 users

- Frame latency peaks up-to 800 and 700ms in the case of 4 and 6 users, respectively.
- Avg. frame latency without the algorithm: approx. 90-100ms
- Avg. frame latency with algorithm: approx. 70-80 milliseconds.



Conclusion and Future Work

- The A-Frame **browser-based multi-user VR application** can be rendered on remote devices with limited processing and display features.
- The **ARS-VR algorithm** was also introduced, aimed at improving VR performance in terms of frame rate and frame latency on such devices, with the dynamic adaptation of the visuals resolution.
- ARS-VR significantly **reduces the FPS loss** in the scenarios of 4, 6, and 8 simultaneous users by 23%, 30%, and 28%, respectively.
- Future work will improve the algorithm to **support devices** with different screen height and width ratios. **Other user applications**, such as video conferencing in A-Frame and other device types can also be studied.



Thank you for your attention

Anderson Simiscuka, Dublin City University

 www.traction-project.eu

Anderson Simiscuka (DCU)
anderson.simiscuka2@mail.dcu.ie
www.eeng.dcu.ie/~pel



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 870610



Traction
Opera co-creation
for a social
transformation