

Summer intern summary report

----by Joshua, Aug 2023

**the order of the report is from the LEAST successful to the MOST successful.*

1. OpenFOAM Simulations for COVID-19 Airborne Transmission

<https://openfoam.org/version/11/>

The decision to utilize OpenFOAM for conducting CFD simulations on COVID-19 airborne transmission stemmed from a thought-provoking article that explored the impact of indoor ventilation on the spread of the virus. Despite encountering a few initial challenges, such as the complexity of mesh generation and the resource-intensive nature of running the simulations, the OpenFOAM pipeline proved to be a promising avenue for investigation.

Running a successful CFD simulation involves a collaborative process that encompasses CAD model design, OpenFOAM model meshing, computational analysis, and the visualization of results using ParaView. However, two distinct hurdles were encountered during this endeavor.

The first challenge involved file conversion. OpenFOAM only accepts CAD files in STL and DTL formats. Unfortunately, working with the AutoCAD 2024 version posed difficulties in exporting accurate STL files, hindering the progress of the simulations. Overcoming this obstacle necessitated additional efforts in troubleshooting and finding suitable workarounds to obtain the required file format.

The second hurdle revolved around setting boundary conditions using code. While the original code provided a starting point, it did not cater to all the specific conditions that needed to be simulated accurately. Modifying the code to accommodate these conditions introduced a level of complexity that affected the meshing process, leading to unexpected failures and posing challenges in achieving the desired simulation outcomes.

Despite these challenges, the potential of the OpenFOAM computing pipeline for COVID-19 simulations remains promising. Although the simulations have not reached their final stages due to various factors, such as time constraints and the need for further refinement, the groundwork laid thus far warrants continued exploration in future endeavors, such as upcoming internships or research projects.

2). Simscale simulations for cov-19 airborne transmission.

<https://www.simscale.com>

Following the initial setbacks encountered during the utilization of OpenFOAM, an alternative approach was pursued, leading to a shift in focus towards SimScale—an online cloud-computing software tailored for CFD simulations, renowned for its user-friendly

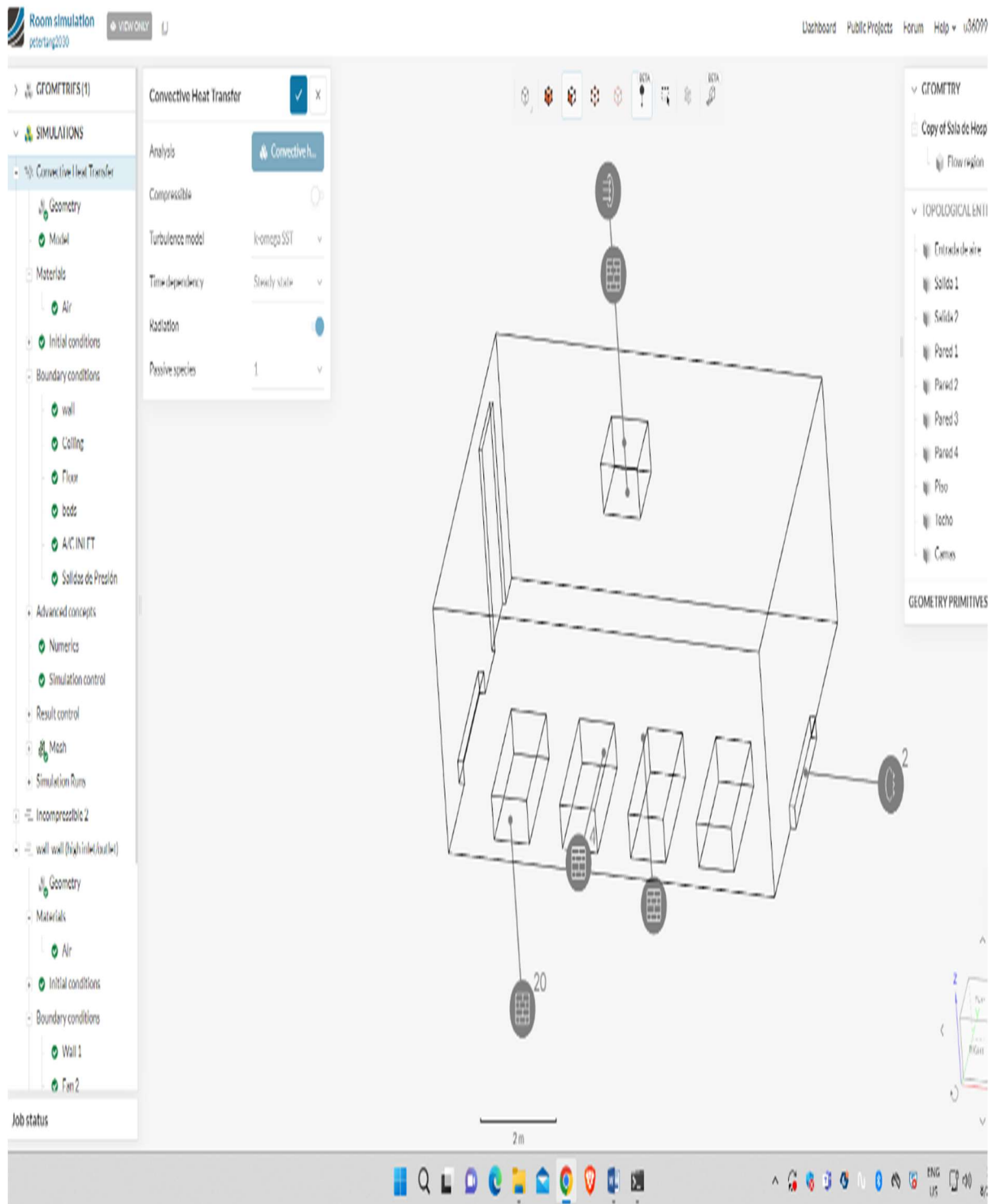
interface and accessibility. SimScale emerged as an intuitive platform, offering a promising solution to overcome the challenges faced earlier, presenting a pathway towards progress.

With SimScale as the newfound ally, the subsequent steps embarked upon a journey of immense potential. The process commenced by seamlessly uploading the DWG file, serving as the foundation for constructing a comprehensive and intricate model, meticulously designed to facilitate simulation. At the core of this endeavor lay a primary objective—to delve deep into the intricacies of convective heat flux, an intricate phenomenon encompassing an analysis of both heat transfer and fluid dynamics within the simulated environment. Undertaking this feat necessitated the configuration of boundary conditions, a crucial task that demanded utmost precision and accuracy in order to faithfully replicate real-world scenarios, thus paving the way for meaningful and impactful outcomes.

Once the intricate web of boundary conditions was meticulously woven, the initiation of a new simulation on SimScale was reduced to a mere matter of minutes. Leveraging the cloud-based nature of the software, the computational processing unfolded with unrivaled efficiency, resulting in a significant reduction in the time required to obtain comprehensive results. Evidently, the approximate runtime of a mere 30 minutes showcased SimScale's prowess as it effortlessly unveiled invaluable insights into the fluid velocity distribution across various sections, spanning the realms of the x, y, and z planes—a testament to its unwavering dedication towards delivering excellence.

Upon a closer examination of the simulation results, a tapestry of intriguing patterns unfurled before our eyes, revealing captivating revelations. Notably, the discerning eye could not overlook the stark disparity in fluid velocity magnitude, particularly in the vicinity of the lower sections of the four walls. In stark contrast to the incoming airflow originating from the air conditioning system and the subsequent outflow, the fluid velocity near the bottom portions exhibited a substantial diminishment. This captivating observation manifested itself through a mesmerizing array of colors, as the vibrant hues of red and yellow seamlessly transitioned into the more tranquil shades of blue—a powerful visual representation of the relatively stagnant regions near the walls' lower extremities. These compelling findings, akin to a treasure trove of knowledge, offer invaluable insights into the optimization of ventilation systems. They illuminate the path towards achieving both effective air circulation and the faithful simulation of particle dispersion, as they shed light on areas that necessitate attention, calling for meticulous adjustments to ensure the seamless integration of efficient airflow within the realm of the simulated environment.

The ability to seamlessly visualize and meticulously analyze fluid dynamics through the powerful simulations facilitated by SimScale unveils a realm of endless possibilities, opening the doors to untapped potential within the domain of indoor design. Harnessing these profound insights, designers and engineers are equipped with the tools to craft ventilation systems that transcend the boundaries of convention. These systems not only ensure the creation of well-ventilated spaces but also mimic the intricate and delicate dance of circulating particles, thus catalyzing the development of safer, more efficient, and meticulously crafted environments that stand as a testament to human ingenuity and progress.



(this graph present the general CAD model we are using for simulation)

Tool and setting at the left

Room simulation
petertang2030

VIEW ONLY

GEOMETRIES

- Copy of Sala de Hospital

SIMULATIONS

- Convective Heat Transfer**
 - Geometry
 - Model
 - Materials
 - Air
 - Initial conditions
 - Boundary conditions
 - wall
 - Ceiling
 - Floor
 - beds
 - A/C INLET
 - Salidas de Presión
 - Advanced concepts
 - Numerics
 - Simulation control
 - Result control
 - Mesh
 - Simulation Runs
- Incompressible 2
- wall-wall (high inlet/outlet)
 - Geometry
 - Materials
 - Air
 - Initial conditions
 - Boundary conditions

Convective Heat Transfer

Analysis **Convective h...**

Compressible ☐

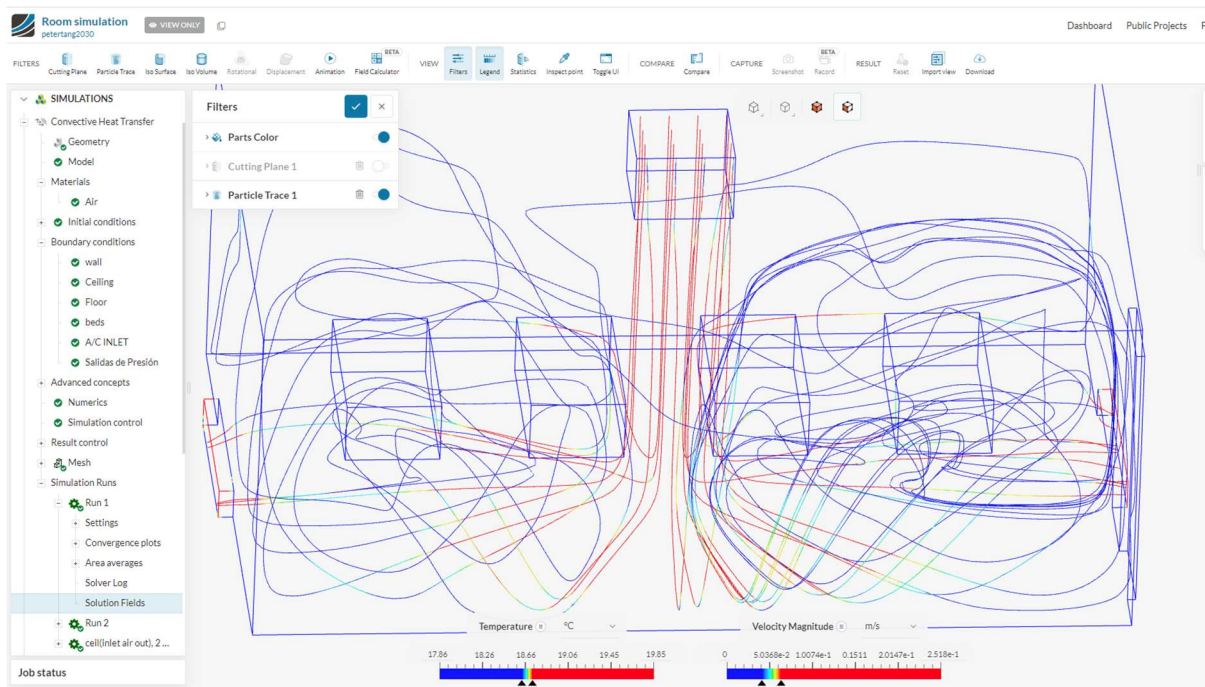
Turbulence model k-omega SST

Time dependency Steady-state

Radiation ☒

Passive species 1

Job status



3. ContolNet stable diffusion via Colab

Once I accomplished the task about the CFD animation, I moved to AI-assisted controlling system via an open sourced pipeline called stable diffusion. Stable diffusion is a deep learning, text-to-image and image-to-image (and ContolNet pipeline is the most useful one amongst) released in 2022 based on diffusion techniques. Some tasks such as inpainting, outpainting, and generating image-to-image translations guided by a text prompt. The model used by stable diffusion is a latent diffusion model, a deep generative artificial neural network, the code and model can run on most consumer hardware equipped with a modest GPU with at least 8GB RAM (for simplicity of downloading difficulties, here I will introduce the cloud computing version though Colab by Google)

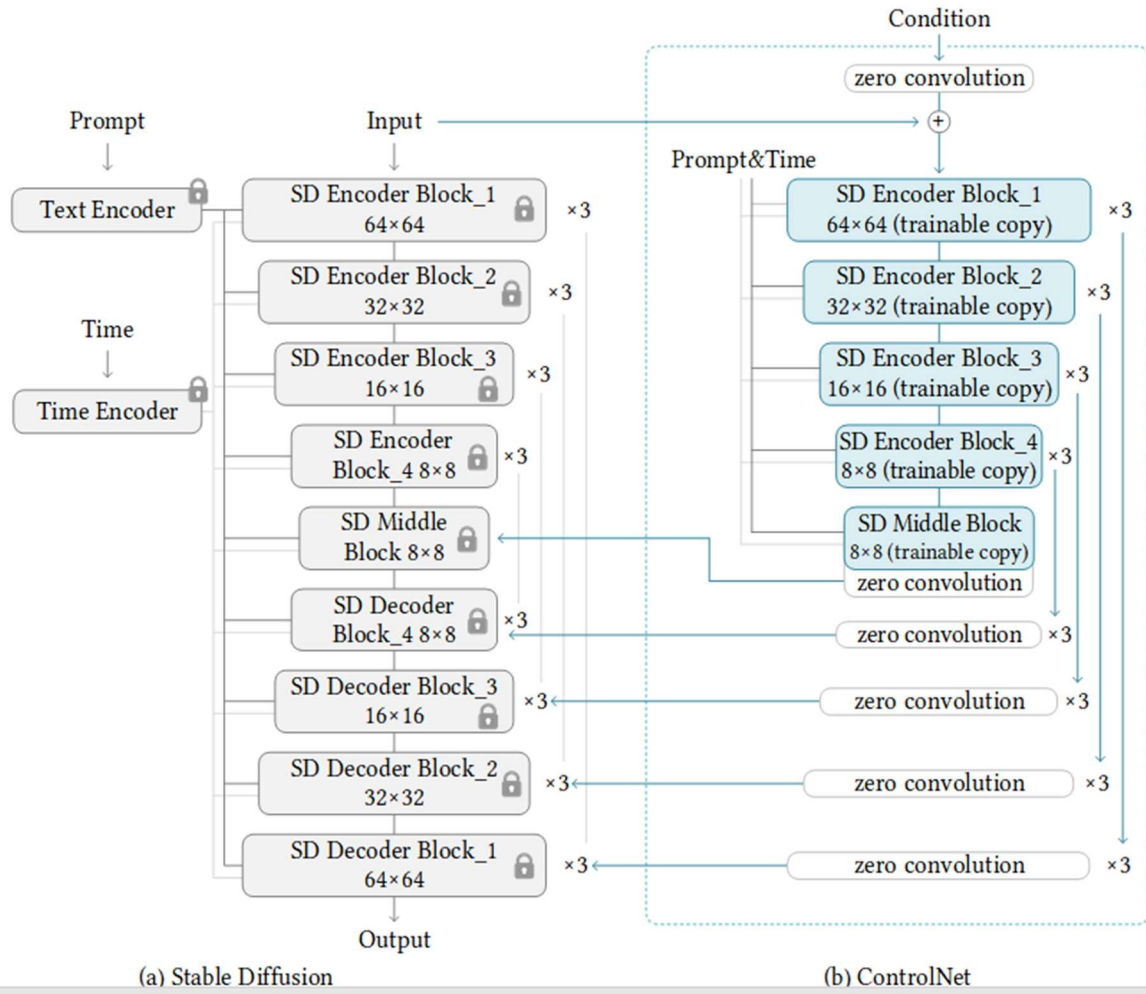
Here is the website for reference:

<https://colab.research.google.com/github/huggingface/notebooks/blob/main/diffusers/controlnet.ipynb>

After entering to this website, we shall login to google account to successfully connect to google service. (top-right corner)

Lets study how stable diffusion works in the following page.


Pictorially, training a ControlNet looks like so:



If you cannot understand these, we can run the code first by click the button (red circle 1-3) shown here BY SEQUENCE.

We welcome you to run the code snippets shown in the sections below with [this Colab Notebook](#).

Before we begin, let's make sure we have all the necessary libraries installed:

1 

```
!pip install -q diffusers==0.14.0 transformers xformers git+https://github.com/huggingface/accelerate.git
```


To process different conditionings depending on the chosen ControlNet, we also need to install some additional dependencies:

- [OpenCV](#)
- [controlnet-aux](#) - a simple collection of pre-processing models for ControlNet

2 

```
!pip install -q opencv-contrib-python
!pip install -q controlnet_aux
```


We will use the famous painting "[Girl With A Pearl](#)" for this example. So, let's download the image and take a look:

3 

```
from diffusers import StableDiffusionControlNetPipeline
from diffusers.utils import load_image

image = load_image(
    "https://hf.co/datasets/huggingface/documentation-images/resolve/main/diffusers/input_image_vermeer.p"
)
image
```

WARNING:xformers:A matching Triton is not available, some optimizations will not be enabled.
Error caught was: No module named 'triton'



Notes that the URL after load_image is the link of photos in your favor and it shall be .png format.

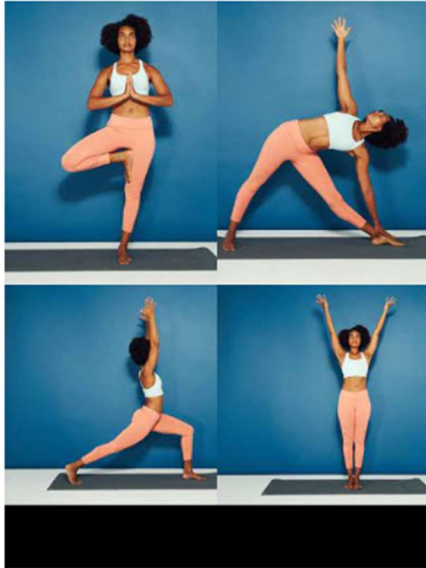
And finally, we come to a stage of image controlling. In the code shown below, there are two line of constrain. The first line of prompt describes the requirement for all four pictures while

the next prompt ask for specific requirement for each individual picture.

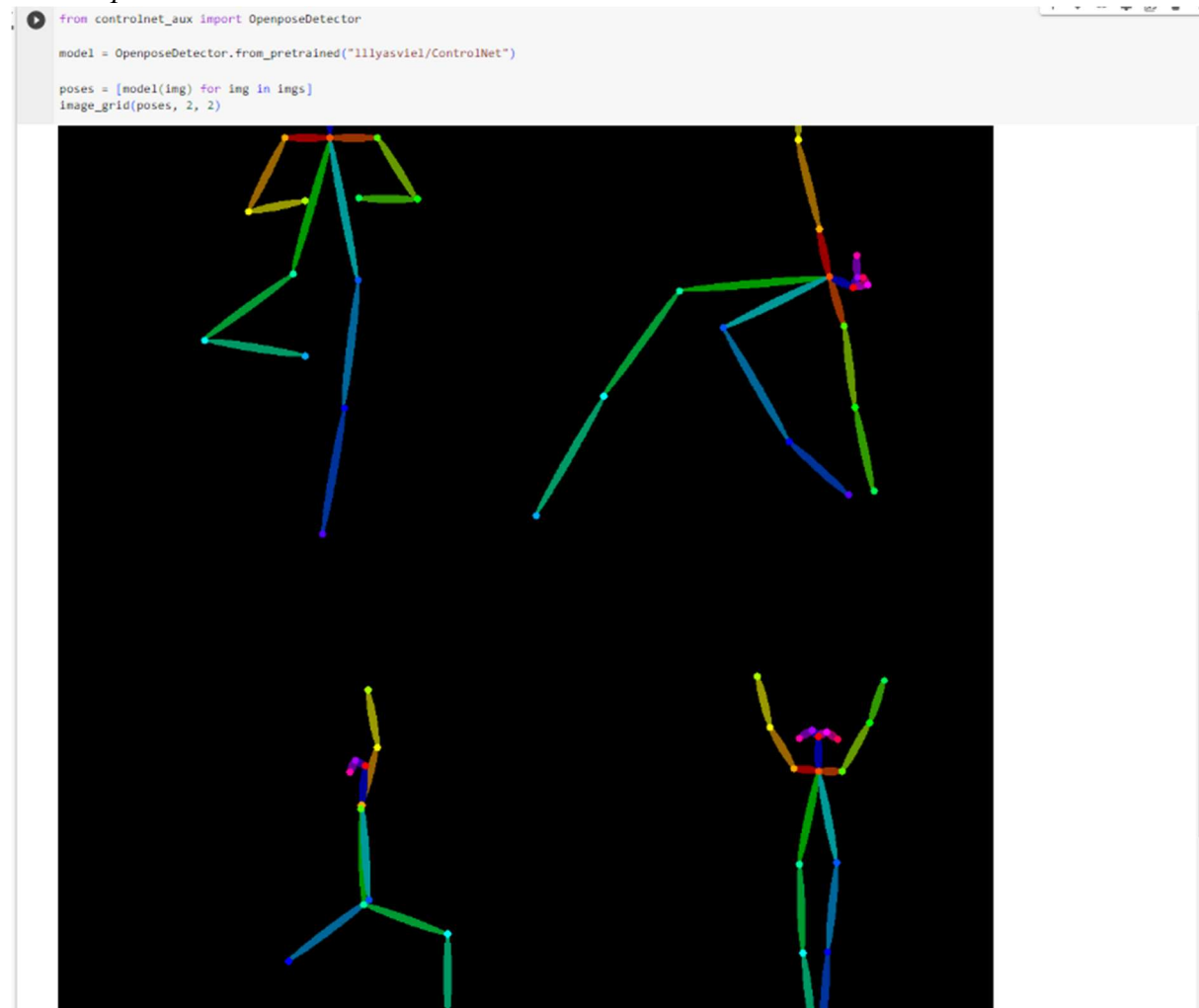


Apart from this application on face replacing, ControlNet also works very well on pose detection and animation using prompt. Here is the four poses we're interested in. and


```
[10] urls = "yoga1.jpeg", "yoga2.jpeg", "yoga3.jpeg", "yoga4.jpeg"
      imgs = [
        load_image("https://hf.co/datasets/yiyixu/controlnet-testing/resolve/main/" + url)
        for url in urls
      ]
      image_grid(imgs, 2, 2)
```



and the poses is detected.



On the finally code about to execute, the negative_prompt refer to some trait we do not need but may exist if we do not include it, for example, worst quality and low quality.

```
generator = [torch.Generator(device="cpu").manual_seed(2) for i in range(4)]
prompt = "super-hero character, best quality, extremely detailed"
output = pipe(
    [prompt] * 4,
    poses,
    negative_prompt=["monochrome, lowres, bad anatomy, worst quality, low quality"] * 4,
    generator=generator,
    num_inference_steps=20,
)
image_grid(output.images, 2, 2)
```

