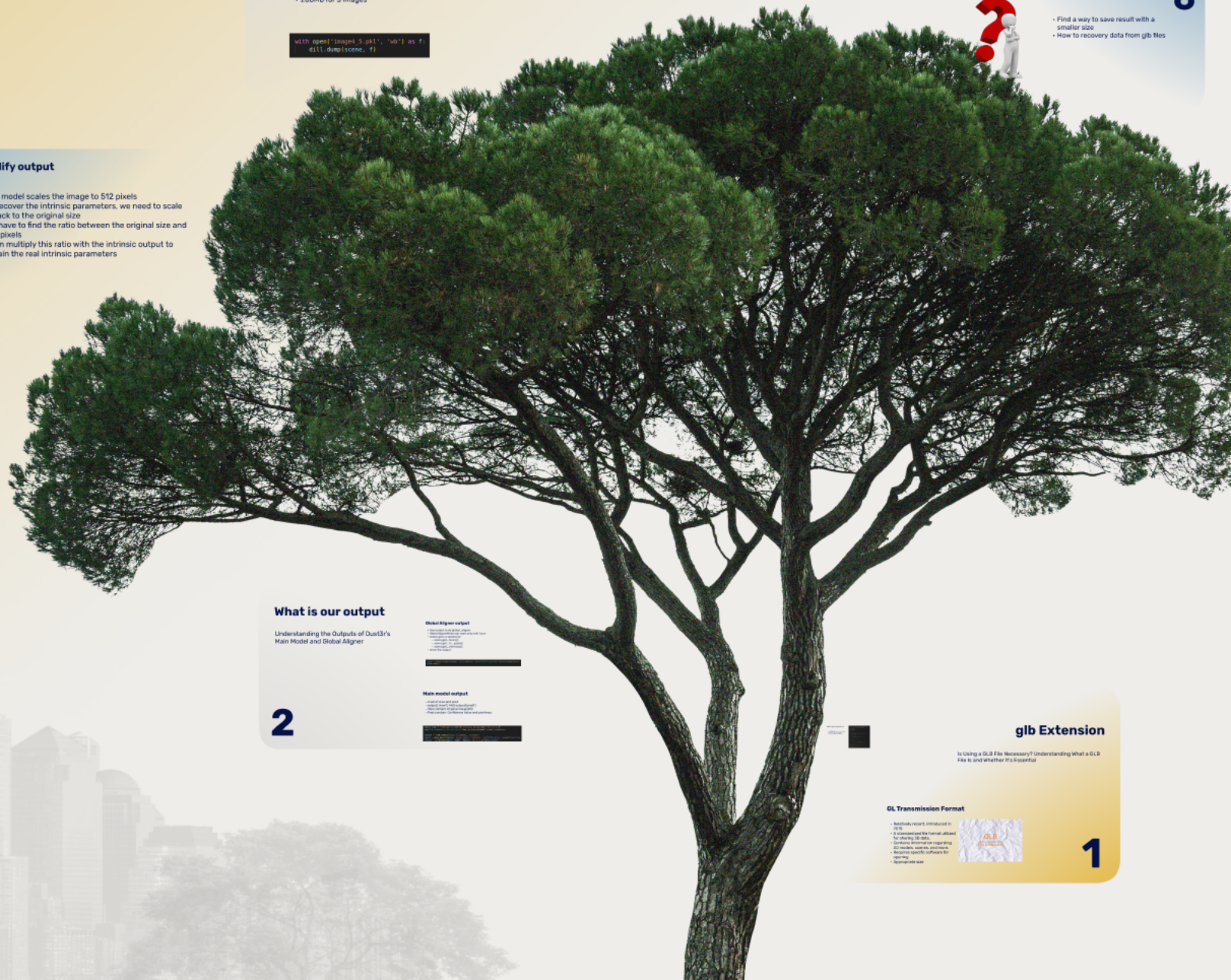
DUSt3R - Second Try

Go deeper with code



glb Extension

Is Using a GLB File Necessary? Understanding What a GLB File Is and Whether It's Essential

GL Transmission Format

- Relatively recent, introduced in 2015
- A standardized file format utilized for sharing 3D data.
- Contains information regarding 3D models, scenes, and more.
- Requires specific software for opening
- Appropriate size

1

GL Transmission Format

- Relatively recent, introduced in 2015
- A standardized file format utilized for sharing 3D data.
- Contains information regarding 3D models, scenes, and more.
- Requires specific software for opening
- Appropriate size

Not compulsory to use it

- In the Dust3r they convert glb for show the result at the end
- Not only way for show result

```
Codeium: Refactor | Explain | Generate Docstring | ×
def _convert_scene_output_to_glb(outdir, imgs, pts3d, mask, focals, cams2world, cam_size=0.05,
                                 cam_color=None, as_pointcloud=False,
                                 transparent_cams=False, silent=False):
    assert len(pts3d) == len(mask) <= len(imgs) <= len(cams2world) == len(focals)
    pts3d = to_numpy(pts3d)
    imgs = to_numpy(imgs)
    focals = to_numpy(focals)
    cams2world = to_numpy(cams2world)

    scene = trimesh.Scene()

    # full pointcloud
    if as_pointcloud:
        pts = np.concatenate([p[m] for p, m in zip(pts3d, mask)])
        col = np.concatenate([p[m] for p, m in zip(imgs, mask)])
        pct = trimesh.PointCloud(pts.reshape(-1, 3), colors=col.reshape(-1, 3))
        scene.add_geometry(pct)
    else:
        meshes = []
        for i in range(len(imgs)):
            meshes.append(pts3d_to_trimesh(imgs[i], pts3d[i], mask[i]))
        mesh = trimesh.Trimesh(**cat_meshes(meshes))
        scene.add_geometry(mesh)

    # add each camera
    for i, pose_c2w in enumerate(cams2world):
        if isinstance(cam_color, list):
            camera_edge_color = cam_color[i]
        else:
            camera_edge_color = cam_color or CAM_COLORS[i % len(CAM_COLORS)]
        add_scene_cam(scene, pose_c2w, camera_edge_color,
                      None if transparent_cams else imgs[i], focals[i],
                      imsize=imgs[i].shape[1::-1], screen_width=cam_size)

    rot = np.eye(4)
    rot[:3, :3] = Rotation.from_euler('y', np.deg2rad(180)).as_matrix()
    scene.apply_transform(np.linalg.inv(cams2world[0] @ OPENGL @ rot))
    outfile = os.path.join(outdir, 'scene.glb')
    if not silent:
        print('(exporting 3D scene to', outfile, ')')
    scene.export(file_obj=outfile)
    return outfile
```

What is our output

Understanding the Outputs of Dust3r's Main Model and Global Aligner

Global Aligner output

- Use output to do global_aligner
- GlobalAlignerMode can work only with 1 pair
- scene give us access to:
  - scene.get_focals()
  - scene.get_im_poses()
  - scene.get_intrinsics()
- show the output

```
scene = global_aligner(output, device=device, mode=GlobalAlignerMode.PointCloudOptimizer)
scene.show()
```

Main model output

- 2 set of view and pred
- output['view1'] AND output['pred1']
- View contain: Original image(2D)
- Pred condain Confidence Value and pointmap

```
model_name = "checkpoints/DUSt3R_ViTLarge_BaseDecoder_512_dpt.pth"
model = AsymmetricCroCo3DStereo.from_pretrained(model_name).to(device)

images = load_images(image_filenames, size=512)
pairs = make_pairs(images, scene_graph='complete', prefilter=None, symmetrize=True)
output = inference(pairs, model, device, batch_size=batch_size)
```

2

Main model output

- 2 set of view and pred
- output['view1'] AND output['pred1']
- View contain: Original image(2D)
- Pred condain Confidence Value and pointmap

```
model_name = "checkpoints/DUSt3R_ViTLarge_BaseDecoder_512_dpt.pth"
model = AsymmetricCroCo3DStereo.from_pretrained(model_name).to(device)

images = load_images(image_filenames, size=512)
pairs = make_pairs(images, scene_graph='complete', prefilter=None, symmetrize=True)
output = inference(pairs, model, device, batch_size=batch_size)
```

Global Aligner output

- Use output to do global_aligner
- GlobalAlignerMode can work only with 1 pair
- scene give us access to:
  - scene.get_focals()
  - scene.get_im_poses()
  - scene.get_intrinsics()
- show the output

```
scene = global_aligner(output, device=device, mode=GlobalAlignerMode.PointCloudOptimizer)
scene.show()
```

3 Modify output

- The model scales the image to 512 pixels
- To recover the intrinsic parameters, we need to scale it back to the original size
- We have to find the ratio between the original size and 512 pixels
- Then multiply this ratio with the intrinsic output to obtain the real intrinsic parameters

4 Save output

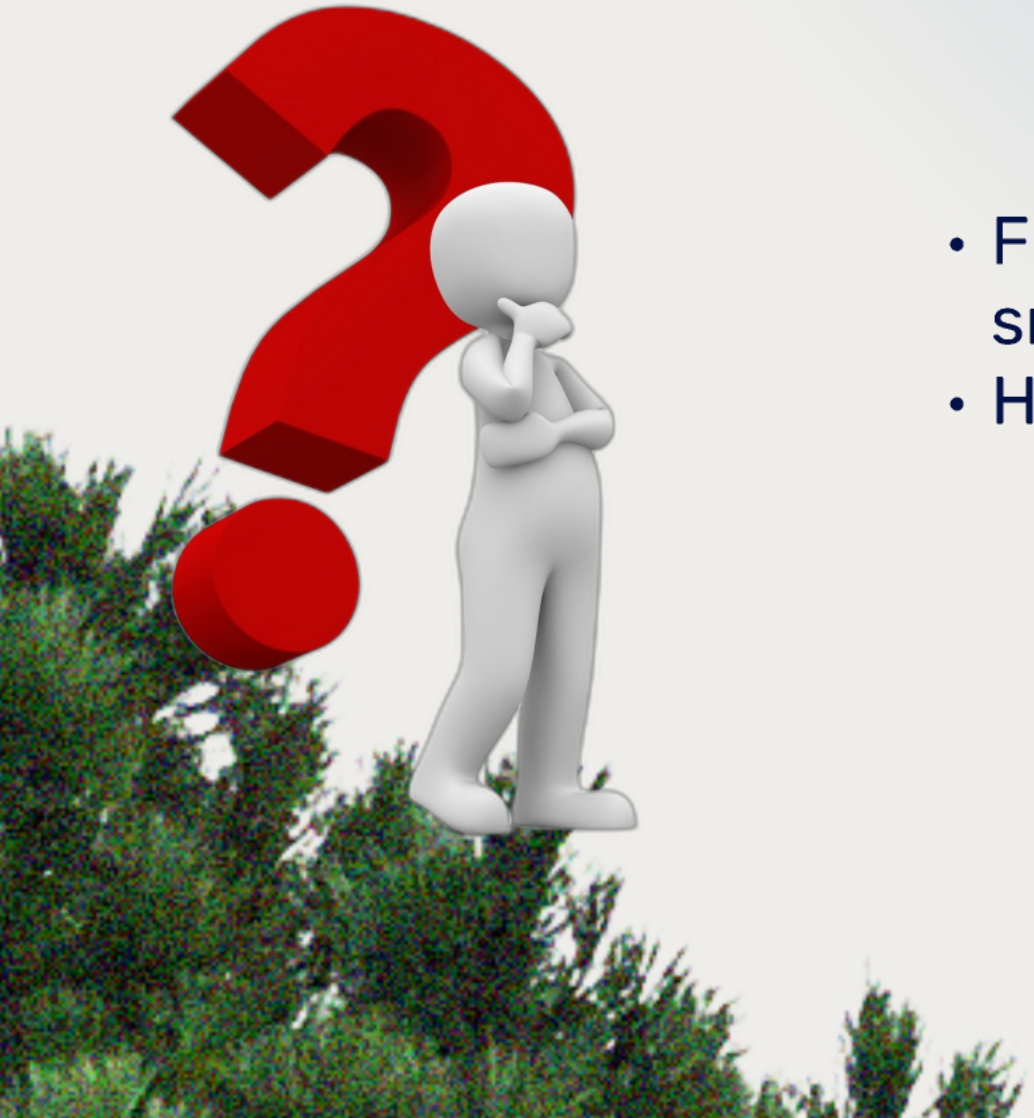
- Save output(scene) using "dill"
- Large size
- 280Mb for 3 images

```
with open('image4_5.pkl', 'wb') as f:
    dill.dump(scene, f)
```

5 Experience with the Sinergia workspace

- With GPU, processing more than around 15 images not be feasible.
- For more images only have to use CPU
- Takes time!!!
- Takes more than 1.5h for 30 images

Future Work 6



- Find a way to save result with a smaller size
- How to recovery data from glb files