CS492D: Diffusion Models and Their Applications

Assignment 5 Session

JAIHOON KIM

Fall 2024
KAIST

Introduction

In Assignment 5, you will implement two applications of diffusion synchronization: ambiguous image and arbitrary-sized image generation.

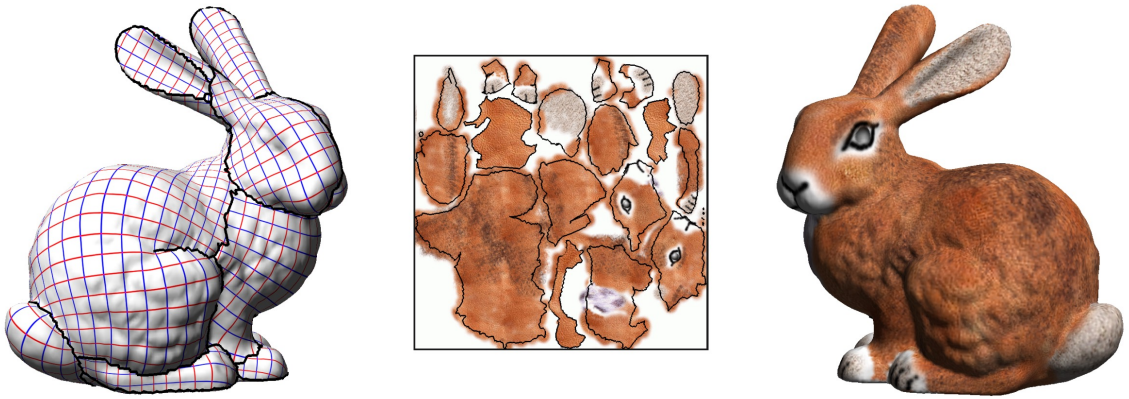
a watercolor painting of an owl

an oil painting of a watering can

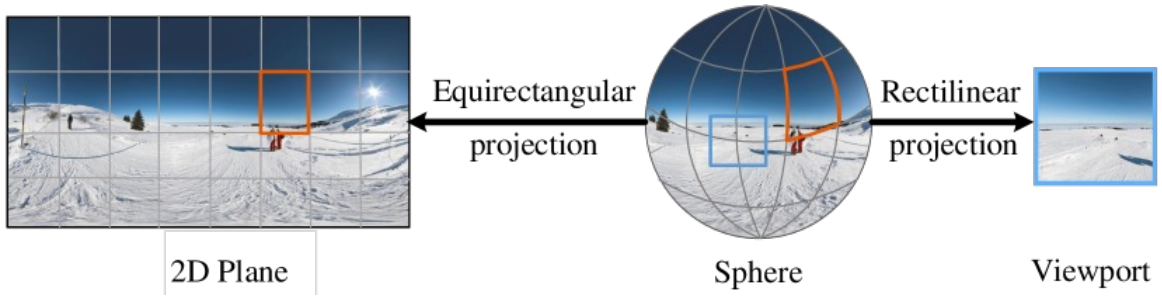
SyncTweedies, Kim *et al.*, NeurIPS 2024

Introduction

Note that one can extend to generate other types of visual content by switching the projection operation.



Mesh Texture - g = Mesh Render
(Levy et al.)

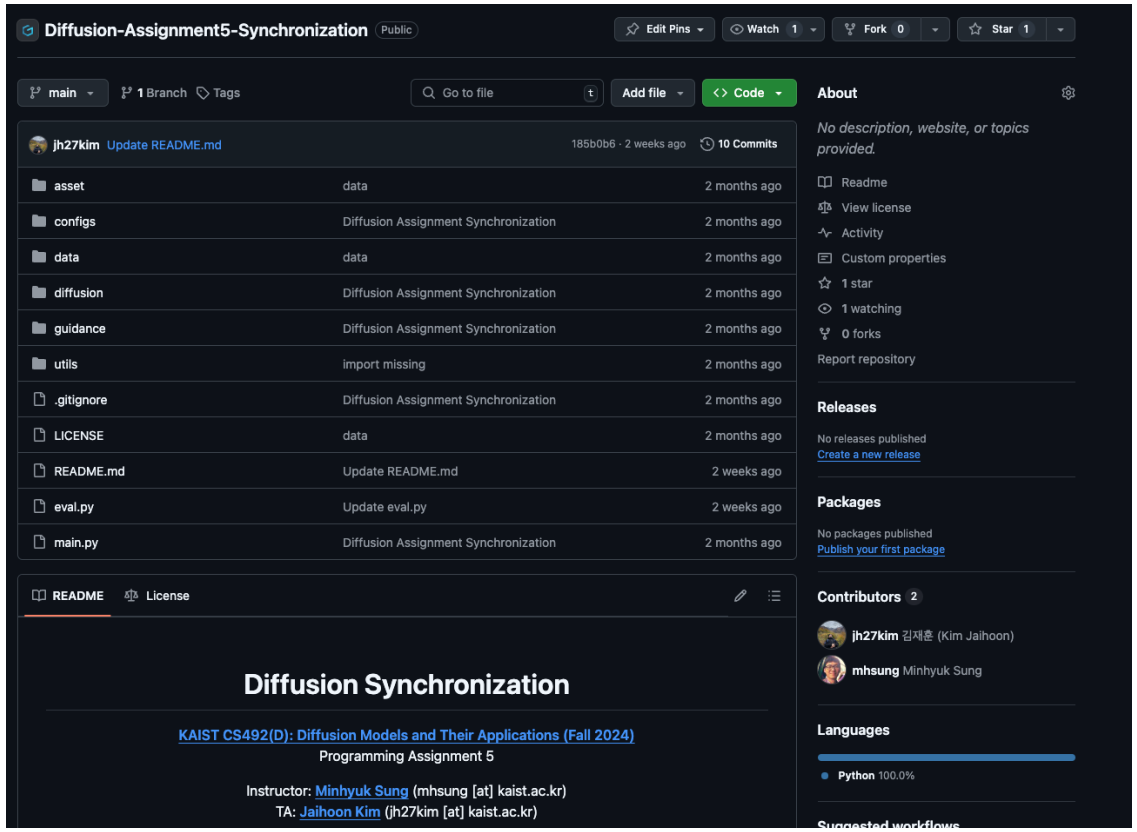


360° Panorama - g = Equirectangular projection
Duan et al.

Introduction

The skeleton code and instructions are available at:

https://github.com/KAIST-Visual-AI-Group/Diffusion-Assignment5-Synchronization

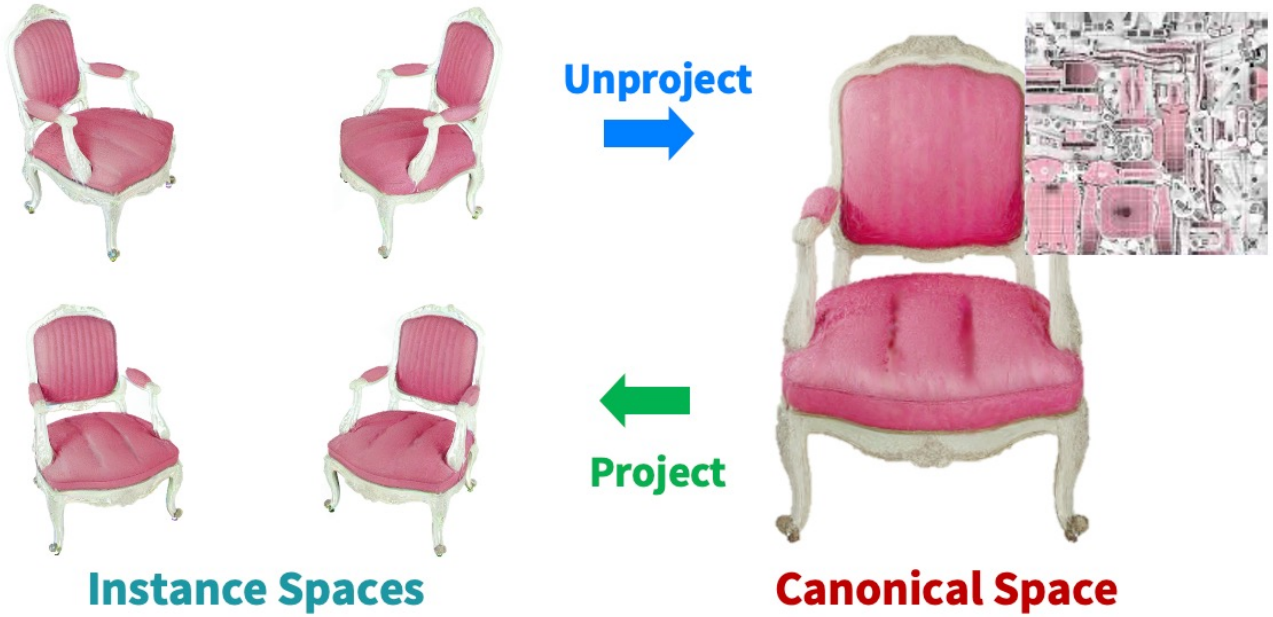


Important Notes

- All programming assignments are due **two weeks** after the assignment session.
- Late submission will incur **20% penalty** for **each** late day!
- Please carefully check the README of each assignment.
- Missing items in your submission will also incur penalties.

What to Do: Overview

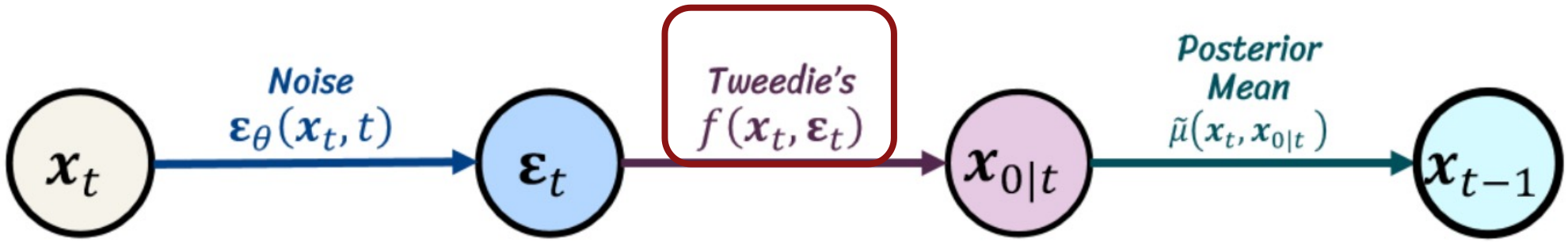
We will focus on implementing the core components of diffusion synchronization: projection and unprojection operation.



SyncTweedies, Kim *et al.*, NeurIPS 2024

What to Do: Task 0

Implement DDIM reverse process ($\sigma_t = 0$ for all t).

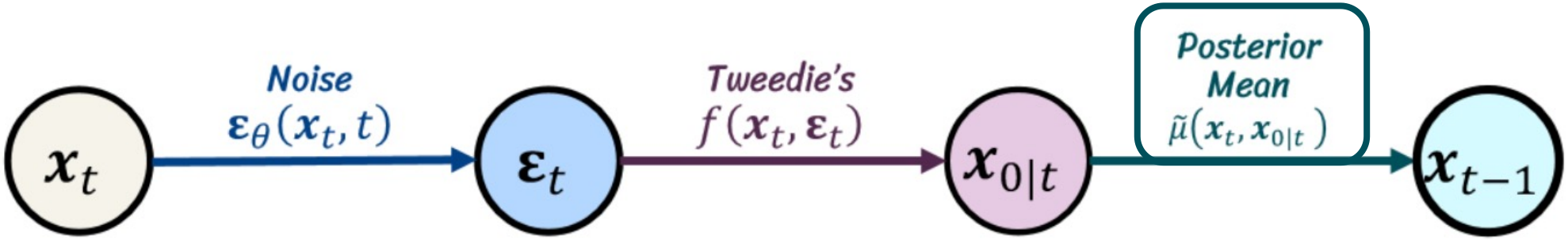


$$x_{0|t} = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \sqrt{1-\bar{\alpha}_t}\hat{\varepsilon}_\theta(x_t, t)\right)$$

`compute_tweedie()` in `guidance/base_model.py`

What to Do: Task 0

Implement DDIM reverse process ($\sigma_t = 0$ for all t).



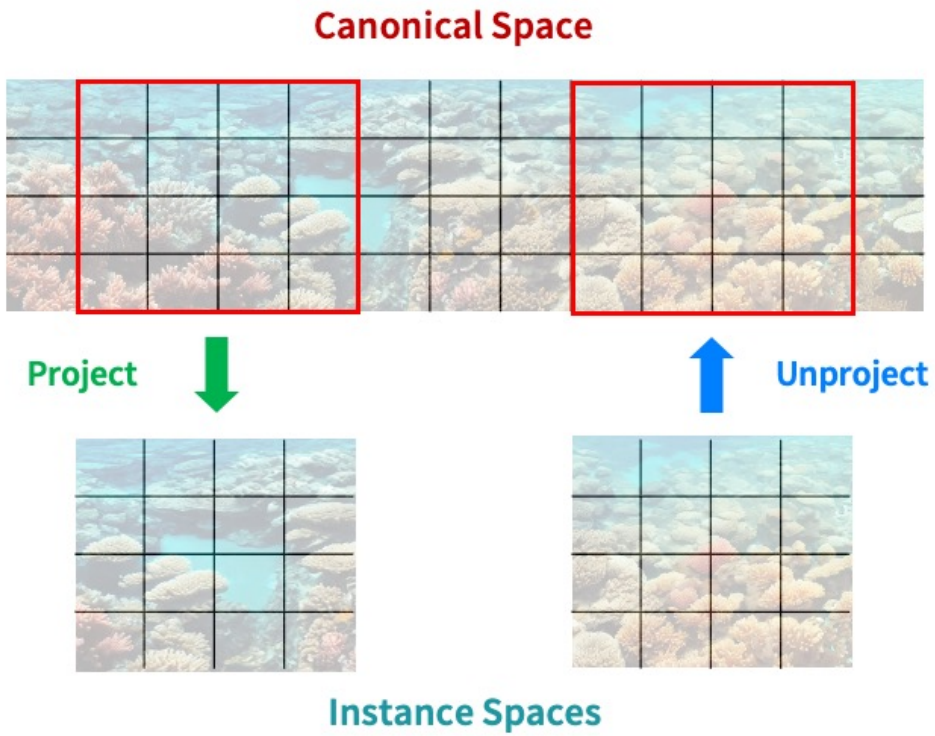
$$\tilde{\mu}(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\boldsymbol{x}_0$$

`compute_prev_state()` in `guidance/base_model.py`

What to Do: Task 1

Implement projection and unprojection of arbitrary-sized image generation.

Projection:
Cropping



Unprojection:
Pasting

What to Do: Task 1

Synchronize the multiple denoising processes by averaging their intermediate $\boldsymbol{x}_{0|t}$.

MultiDiffusion, Bar-Tal *et al.*, ICML 2023

What to Do: Task 1

Input

Prompt: y

"A photo of a mountain range at twilight"

Target image $\boldsymbol{x_0}$

📌 Prompt **alignment**

📌 Global **coherence**

Output

What to Do: Task 1

Initialization of pre-trained diffusion model and text embeddings are given.

Implement TODOs in `guidance/wide_image_model.py`:

- `forward_mapping()`: projection operation
- `inverse_mapping()`: unprojection operation

🎯 Hint: Use `self.mapper`

What to Do: Task 1

Run the following code to generate arbitrary-sized images:

```
python main.py --app wide_image --prompt "{$PROMPT}" --tag wide_image
```

An example of the generated output.

“A desert landscape with vast dunes of golden sand”

What to Do: Task 1

For evaluation, we will crop images at random positions and measure their text alignment using CLIP score.

...

What to Do: Task 1

Use the prompts provided in `data/wide_image_prompts.json` and gather the generated images in a directory.

!! Note that the filenames must match their corresponding text prompts.

```
./
├── A_beach_scene_at_sunrise_with_golden_sands_1005.png
├── A_beach_scene_at_sunrise_with_golden_sands_1199.png
├── A_beach_scene_at_sunrise_with_golden_sands_1669.png
├── A_beach_scene_at_sunrise_with_golden_sands_2220.png
├── A_beach_scene_at_sunrise_with_golden_sands_2292.png
├── A_beach_scene_at_sunrise_with_golden_sands_2300.png
├── A_beach_scene_at_sunrise_with_golden_sands_243.png
├── A_beach_scene_at_sunrise_with_golden_sands_991.png
├── A_bustling_city_skyline_at_night_with_skyscrapers_1141.png
├── A_bustling_city_skyline_at_night_with_skyscrapers_1667.png
├── A_bustling_city_skyline_at_night_with_skyscrapers_1797.png
├── A_bustling_city_skyline_at_night_with_skyscrapers_1915.png
```

What to Do: Task 1

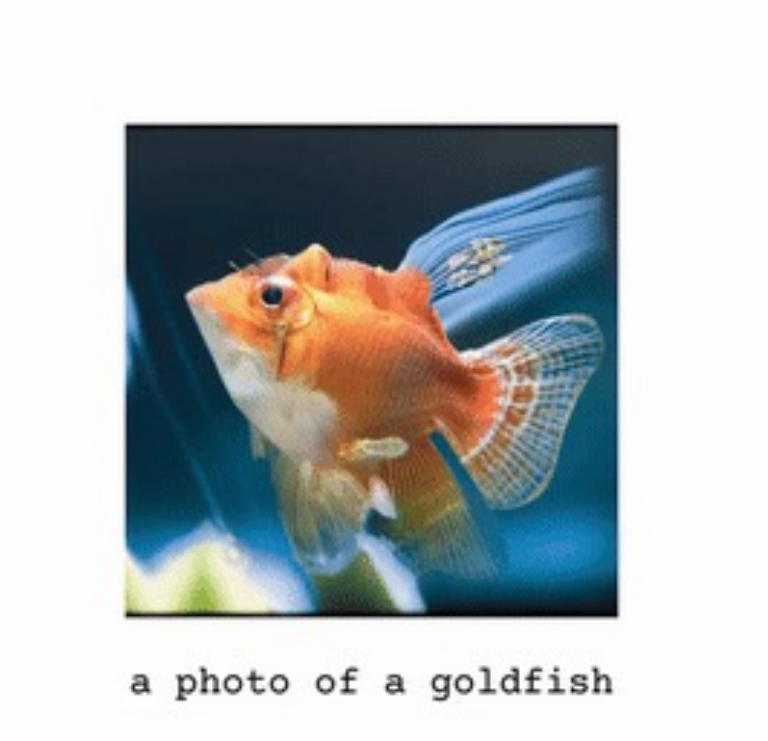
Run the evaluation script, which outputs an `eval.json` file.

```
Python eval.py --fdir1 {$FDIR1} -app wide_images
```


```
{
    "source_dir": "$HOME/",
    "$HOME/A_beach_scene_at_sunrise_with_golden_sands_2292.png": 0.3110335171222687,
    "$HOME/A_photo_of_a_beautiful_ocean_with_coral_reef_486.png": 0.30322808027267456,
    "$HOME/A_tranquil_lake_surrounded_by_misty_mountains_752.png": 0.3230690062046051,
    "$HOME/A_desert_landscape_with_vast_dunes_of_golden_sand_791.png": 0.3255298137664795,
    "$HOME/A_lavender_field_in_Provence_with_rows_of_vibrant_purple_flowers_510.png": 0.3300771117210388,
    "$HOME/A_European_medieval_town_with_narrow_cobblestone_streets_2189.png": 0.31440305709838867,
    "$HOME/A_bustling_city_skyline_at_night_with_skyscrapers_1667.png": 0.2883412837982178,
    "$HOME/A_dense_forest_in_autumn_with_trees_in_shades_of_red_orange_676.png": 0.33474230766296387,
    "$HOME/A_sweeping_panoramic_view_of_the_Grand_Canyon_at_sunset_646.png": 0.33279505372047424,
    "$HOME/The_Great_Wall_of_China_snaking_across_rugged_mountains_147.png": 0.30410248041152954,
    "score": 0.31673217117786406
}
```

What to Do: Task 2

Ambiguous images are images that show different appearances based on transformations.



Visual Anagrams, Geng *et al.*, CVPR 2024

What to Do: Task 2

In this assignment, we focus on 2-view ambiguous images where the projection functions are an identity and a 90° rotation transformation.

"A photo of a goldfish"

"A photo of the queen of the England"

Canonical Space

Instance space

g = Identity

$g = 90^\circ$ Rotation

Visual Anagrams, Geng *et al.*, CVPR 2024

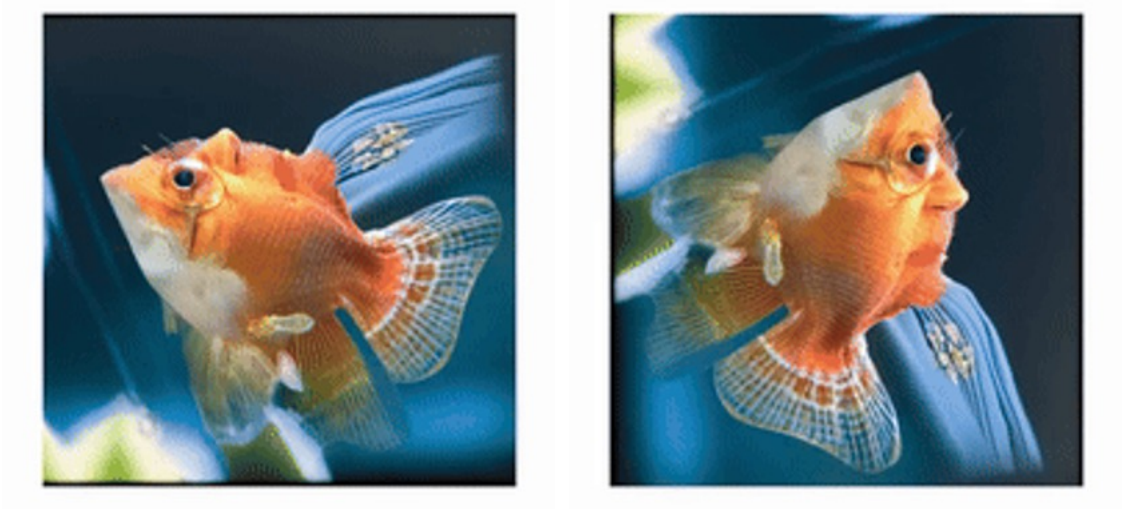
What to Do: Task 2

Input

Prompt: y

"A photo of {a goldfish, the queen of the England}"

Output



Target image $\boldsymbol{x_0}$

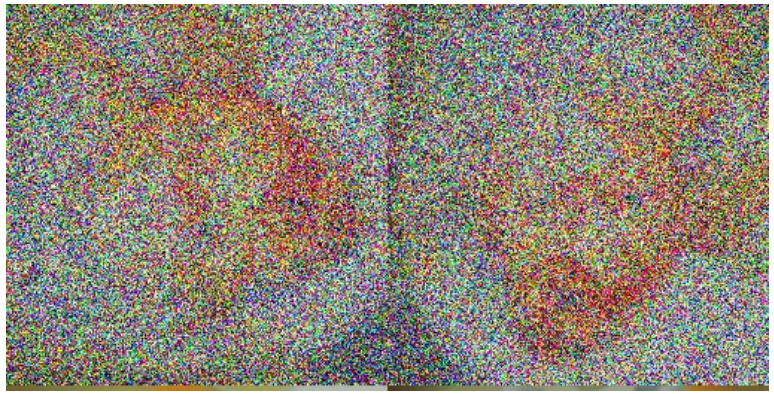
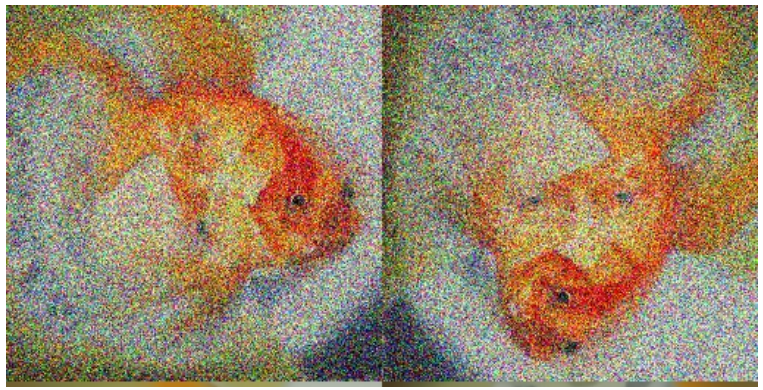
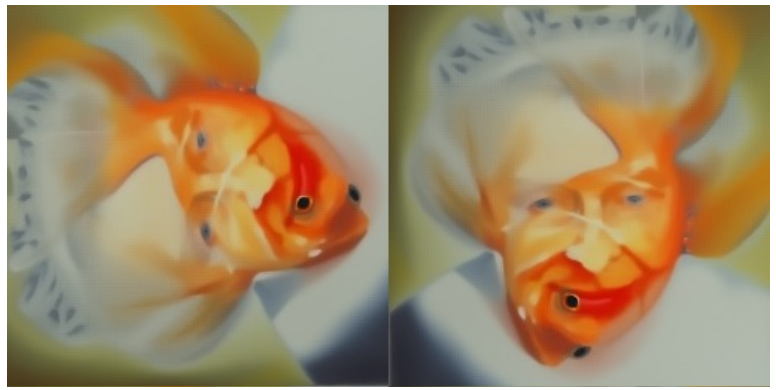
📌 Prompt **alignment**

📌 Global **coherence**

Visual Anagrams, Geng *et al.*, CVPR 2024

What to Do: Task 2

Synchronize the multiple denoising processes by averaging their intermediate $\boldsymbol{x}_{0|t}$.

What to Do: Task 2

Initialization of pre-trained diffusion model and text embeddings are given.

Implement TODOs in `utils/views/view_rotate.py`:

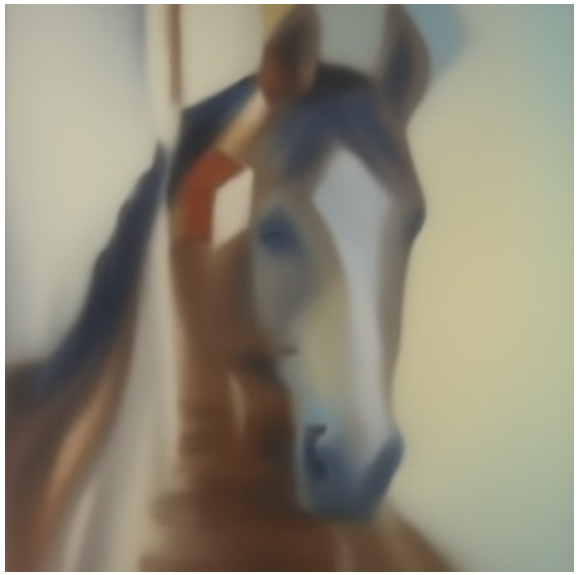
- `view(): projection operation (depends on which image you set as the canonical space)`
- `inverse_view(): unprojection operation`

What to Do: Task 2

Run the following code to generate arbitrary-sized images:

```
python main.py --app ambiguous_image --tag ambiguous_image
```

An example of the generated output.



“An oil painting of a {horse, snowy mountain village}”

What to Do: Task 2

As in task 1, we will measure the text alignment using CLIP score for evaluation.

Use the prompts provided in `data/ambiguous_image_prompts.json`.
Then place the generated images in the same directory.

!! Note that the filenames must match their corresponding text prompts.

```
./
├── a_lithograph_of_a_table.png
├── a_lithograph_of_a_waterfall.png
├── an_oil_painting_of_a_canyon.png
├── an_oil_painting_of_a_horse.png
├── an_oil_painting_of_a_library.png
├── an_oil_painting_of_a_maine_coon.png
├── an_oil_painting_of_a_snowy_mountain_village.png
├── an_oil_painting_of_a_theater.png
├── a_painting_of_a_deer.png
├── a_painting_of_a_dog.png
├── a_painting_of_a_horse.png
├── a_painting_of_a_truck.png
├── a_pencil_sketch_of_flower_arrangements.png
```

What to Do: Task 2

Run the evaluation script, which outputs an `eval.json` file.

```
Python eval.py --fdir1 {$FDIR1} -app ambiguous_images
```


```
{
    "source_dir": "$HOME/",
    "$HOME/an_oil_painting_of_a_snowy_mountain_village.png": 0.24268795549869537,
    "$HOME/a_pencil_sketch_of_flower_arrangements.png": 0.22505266964435577,
    "$HOME/a_lithograph_of_a_table.png": 0.287808358669281,
    "$HOME/a_photo_of_Pisa_tower.png": 0.3175886869430542,
    "$HOME/an_oil_painting_of_a_horse.png": 0.32105761766433716,
    "$HOME/a_lithograph_of_a_waterfall.png": 0.3440597355365753,
    "$HOME/a_painting_of_a_deer.png": 0.32175421714782715,
    "$HOME/a_painting_of_a_horse.png": 0.3311886489391327,
    "$HOME/an_oil_painting_of_a_library.png": 0.33427464962005615,
    "$HOME/a_photo_of_a_car.png": 0.24487349390983582,
    "$HOME/a_photo_of_a_rabbit.png": 0.2947796881198883,
    "$HOME/a_pencil_sketch_of_students_in_a_classroom.png": 0.3398208022117615,
    "$HOME/a_photo_of_the_queen_of_England.png": 0.28319376707077026,
    "$HOME/a_photo_of_a_duck.png": 0.2831934094429016,
    "$HOME/a_painting_of_a_truck.png": 0.29047542810440063,
    "$HOME/an_oil_painting_of_a_theater.png": 0.26503604650497437,
    "$HOME/a_photo_of_a_goldfish.png": 0.3284745216369629,
    "$HOME/an_oil_painting_of_a_maine_coon.png": 0.24546024203300476,
    "$HOME/an_oil_painting_of_a_canyon.png": 0.2698614299297333,
    "$HOME/a_painting_of_a_dog.png": 0.2967958450317383,
    "score": 0.29337186068296434
}
```

What to Submit

Include the following items into a PDF file: `{NAME}_{SID}.pdf`.

Task 1

- Generated arbitrary-sized images with their corresponding prompts.
- A screenshot of `eval.json` showing the CLIP scores for each item and the overall averaged score.

Task 2

- Generated ambiguous images with their corresponding prompts.
- A screenshot of `eval.json` showing the CLIP scores for each item and the overall averaged score.

What to Submit

Create a single ZIP file `{NAME}_{SID}.zip` including:

- The PDF file, formatted according to the guideline;
- Your implemented code.

Your score will be deducted by 10% for *each* missing item.

Please check carefully!

Grading

You will receive up to 20 points from this assignment.

For Task 1 and Task 2, respectively, you will receive:

- 10 points if CLIP score greater than 0.28,
- 5 points if CLIP score greater, or equal to 0.26 and less than 0.28,
- 0 point if CLIP score less than 0.26.

Grading

Additionally, you can optionally receive a maximum of 5 points for Task 3.

Task 3

- 5 points: Achieve CLIP score greater than 0.28.
- 2.5 points: Achieve CLIP score greater, or equal to 0.26 and less than 0.28.
- 0 point: CLIP score less than 0.26.

Thank You