Final Project

CS4365 Applied Image Processing

2023/2024

Course assessment

Assignments 55% (minimal grade 5.0)

- Assignment 1: HDR (33%)
- Assignment 2: Neural image editing (33%)
- Assignment 3: Image warping (33%)

Final project 45% (minimal grade 5.0)

- Implementation
- Report
- Video

Course is passed if the weighted average is bigger or equal 6.0 (rounded on .5).

Final project

- An **individual** project solved during Q1.
- **Goal:** Implement a small application utilizing image processing.
- **Content: 3 components** submitted together (implementation, report and video).
- **Topics:**
  - **Standard:** We provide topics to choose from.
  - **Custom:** You can propose a custom topic of equal or higher perceived complexity.

Feature points

- **Basic features**:
  - Up to 6.0 points.

+ **Extended features**
  - Additional points listed for each topic.

  + ***Custom features***
    - Can compensate other missing features and errors.
    - Must be clearly explained in the video and figures.
    - Points determined by the grader.
    - Only after the extended points.

Max 6 total

Grading

$$\text{Grade} = \text{clamp}(\sum \text{Feature points}, 1, 10)$$

Partial points

- What is not shown does not exist => 0 points.
  - The report and/or video must make it clear that the feature is implemented and that it works correctly.

- Partially implemented / broken / poorly presented features give **partial points**.

  - **Example**:
    - **Feature**: “Resize and letterbox an image to fit the screen.” (1 point)

    - **Solution A**: Only works correctly for square images, distorts others.
    - **Result A**: 1 x 50% = 0.5 points

    - **Solution B**: Produces an image but (almost never) fits the screen.
    - **Result B**: 1 x 0% = 0.0 points

Fraud Policy

- Projects are individual.
  It is not allowed to use external code other than explicitly allowed.

- Copying code, sharing code
  or discussing solutions might be considered fraud.
  (https://www.tudelft.nl/studenten/rechtspositie/fraude-plagiaat/fraud-and-consequences)

Standard Topics

Computational Depth-of-Field

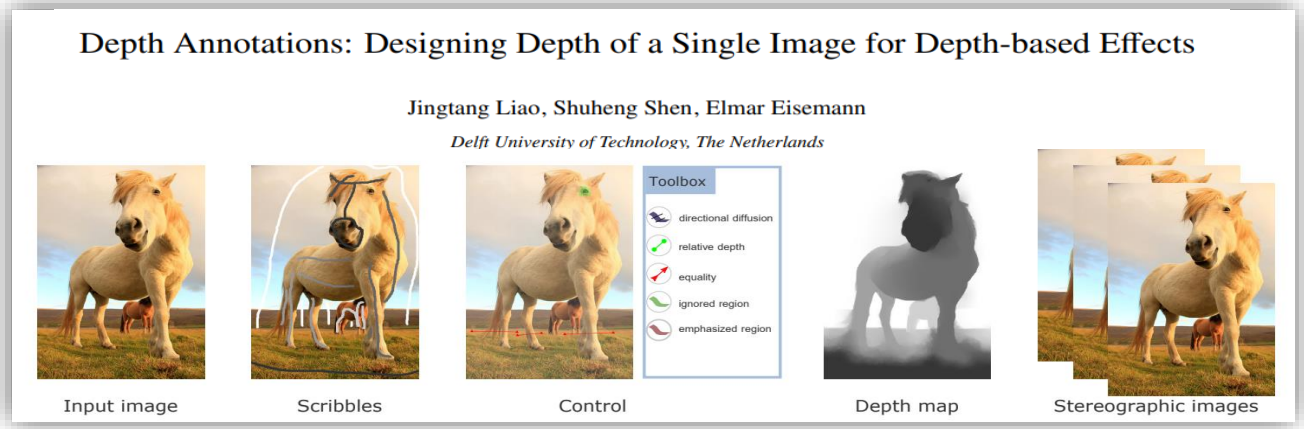
- **Basic features** (6.0):
  1. *Load an RGB image from disk.*
  2. Allow users to scribble depth annotations in UI.
  3. Diffuse annotations across the image using **Poisson image editing**.
  4. Allow users to select focus depth and aperture size.
  5. Simulate depth-of-field using a spatially varying cross-bilateral filter.
  6. *Save and display the result.*
- **Extended features**:
  - Use *a pretrained RGB->Depth CNN* to supplement the depth (+1.0)
  - Find a user-friendly way to combine predicted depth map and user scribbles (+1.0)
  - Implement Ken-Burns effect with depth-based parallax (+2.0)

See notes on the next slide!

Steps in green cursive can be implemented using a 3rd party library.

Clarifications: Poisson image editing (step 3)

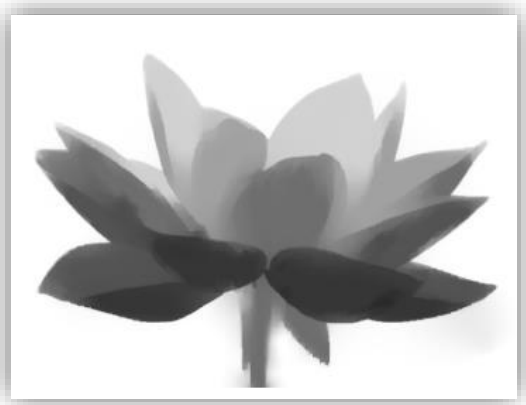
- The basic algorithm [1] is a starting point but alone it will not produce high-quality results.
- You should propose your own solution building upon the gradients.
- The **Depth Annotations** paper [2] is a possible direction:
  - https://graphics.tudelft.nl/Publications-new/2017/LSE17a/depthannotations-authorsversion.pdf



[1] Pérez, Patrick, Michel Gangnet, and Andrew Blake. "Poisson image editing." *ACM Transactions on Graphics (TOG)* 22.3 (2003): 313-318.

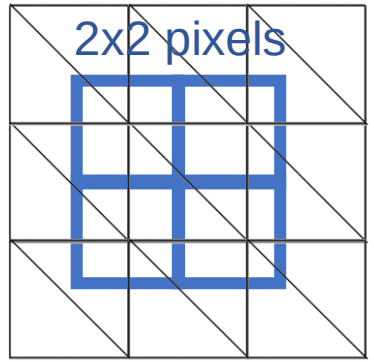
[2] Liao, Jingtang, Shuheng Shen, and Elmar Eisemann: "Depth annotations: Designing depth of a single image for depth-based effects." Computers & Graphics (2018).

Input + scribble → **Monocular depth + focus point** → **Computational depth of field**

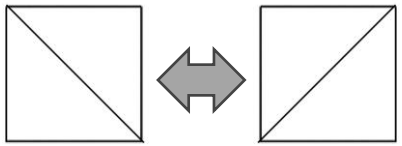


Seam-Carved Vectorization

Step 6:



- **Basic features** (6.0):
  1. *Load an RGB image from disk.*
  2. Run *a pre-trained CNN* for image detection or object classification.
  3. Extract *feature map from a CNN using Grad-CAM* (https://arxiv.org/abs/1610.02391)
  4. Modify the feature map by painting.
  5. Use the map as a guide for seam carving and remove pixel columns with low values.
  6. Vectorize the remaining pixels by replacing them by triangle pairs.
     - Connect centers of 4 neighboring pixels to define 2 triangles.
  7. Move the vectors back to their original positions by “uncarving” the previously removed columns (you need to remember which ones these were).
  8. Smoothly interpolate the colors in the now stretched vector graphics and rasterize it back to an image.
  9. *Save and display the result.*
- **Extended features**:
  - Visualize the steps of the carving (Step 5) (+1.0)
  - Add *another CNN with features* conditioned on a different type of user input (text, depth map, image, sketch,…) (+2.0)
  - Devise a strategy for orientation of the triangle diagonals (+1.0).

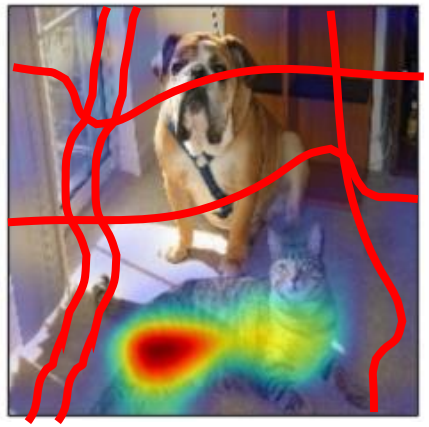


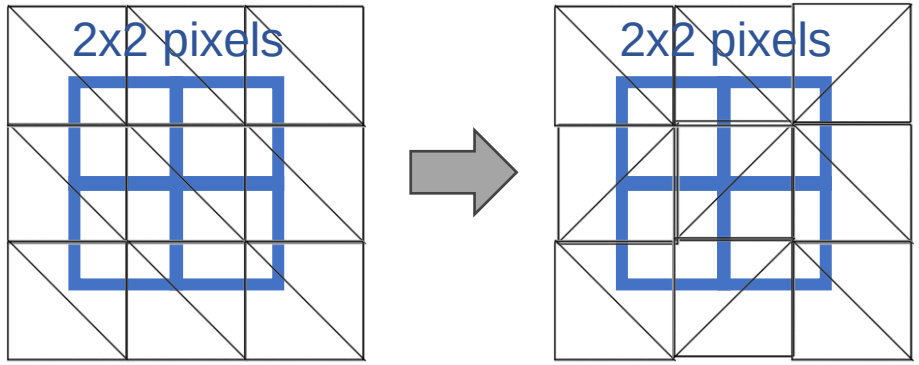
Steps in green cursive can be implemented using a 3rd party library.

Added on October 11, 2023.

Clarifications: Triangles & Interpolation

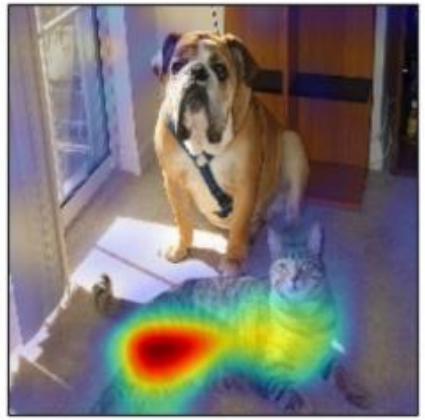
The extension #3 (*"Devise a strategy for orientation of the triangle diagonals"*) can be impactful only if the "uncarved" regions are not strictly one-dimensional. That could be achieved by carving in both vertical and horizontal directions to create more complexly shaped gaps in the image.



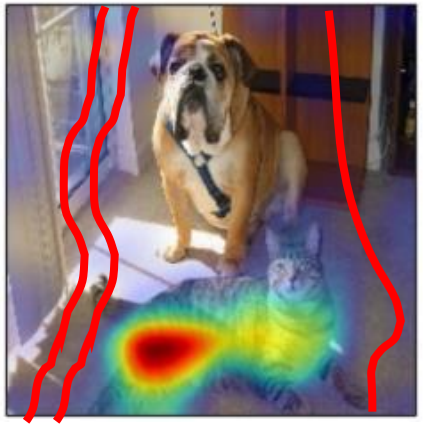


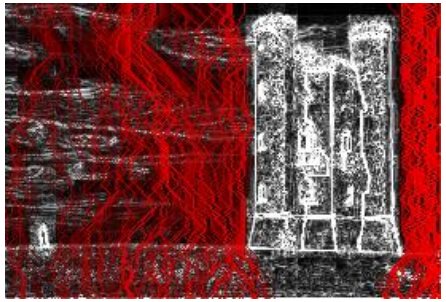
Input → **Grad-CAM** → **Seam Carving** → **Vectorization** → **“Uncarving”**

(a) Original Image



(c) Grad-CAM ‘Cat’





- Selvaraju, Ramprasaath R., et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization." *Proceedings of the IEEE international conference on computer vision*. 2017.
- https://en.wikipedia.org/wiki/Seam_carving
- Dziuba, Maria, et al. "Image Vectorization: a Review." *arXiv preprint arXiv:2306.06441* (2023).

Face Morphing

- **Basic features** (6.0):
  1. *Load 2 RGB images of two faces from disk.*
  2. Run *a pre-trained face landmark* detector on both images.
  3. Allow user to edit/add/remove the landmarks (UI).
  4. Interpolate the landmark **positions and colors** from both images (create a morphing sequence for the landmarks alone). Note: Results may be poor without adding more landmark points in Step 3.
  5. Complete the remaining pixels using Shephard interpolation (IDW).
  6. Project the image to a *pretrained face GAN (e.g., using GAN inversion or Pivotal tuning https://arxiv.org/pdf/2101.05278.pdf)* to improve the image.
  7. Repeat steps 4-7 for the entire morphing sequence.
  8. *Save the result (image sequence, video or GIF).*

See the next slide for explanation.

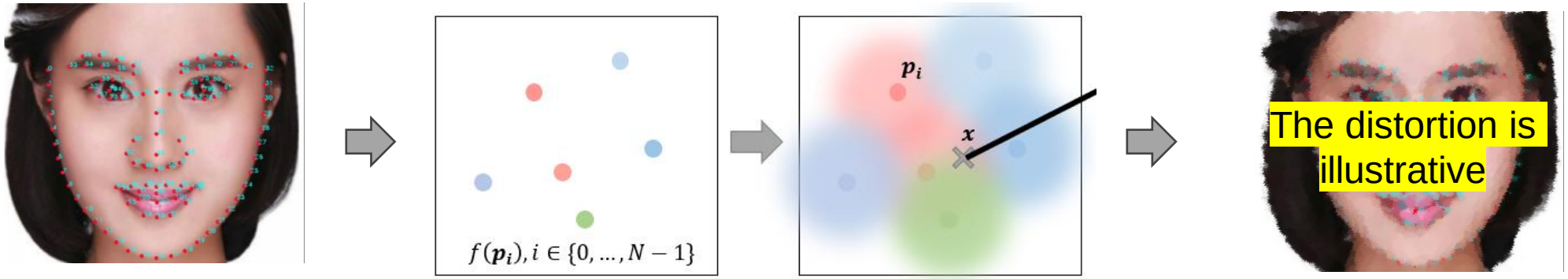
- **Extended features**:
  - Automatically densify the landmarks in Step 3 using consistent *triangulation* and mesh subdivision in both images. (+1.0)
  - Support for objects other than faces (select *suitable features*) (+1.0)
  - Transfer motion from a video (use *an optical flow estimator* and move the landmarks based on that) (up to +2.0)

Steps in green cursive can be implemented using a 3rd party library.

Added on October 10, 2023.

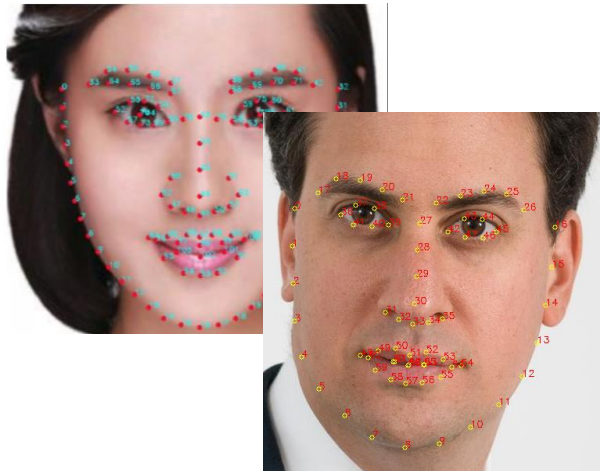
Clarifications: Point interpolation

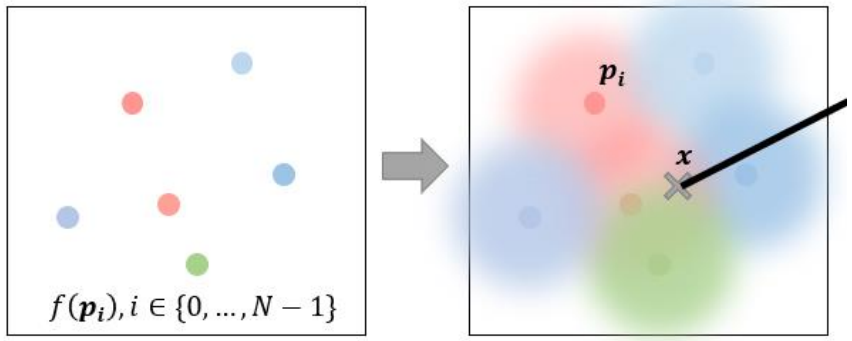
- A **direct** interpolation of colors will lead to a suboptimal quality.
- The colors should instead be interpolated **indirectly**.
- Similarly, as in **mesh-based warping** (Lecture 6):
  1. Morph the landmark **positions**.
  2. Inpaint the position fields with **Shephard interpolation** (choose *q* param).
  3. Use the positions **to backward sample colors** from the original images.

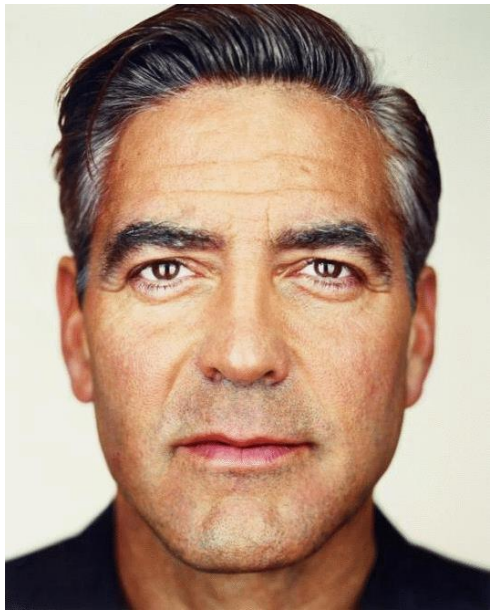


Input | Morphed positions | Interpolated field | Resampled image

Detect landmarks → **Morph & Interpolate** → **Project to GAN** → **Morphing Sequence**







Questions about instructions

- **Not by e-mail**
  - Questions relevant to many are inefficient to answer individually.
- Use public channels:
  - During any lab session.
  - answers.tudelft.nl: https://answers.ewi.tudelft.nl/categories/1/tags/1026
    - Use tag `cs4365`
- You do not need to tell us ahead of time which **standard** topic you chose.

Custom Topics

Custom topics

- Must be of **equal or higher complexity** than standard topics.
  - Estimated by us based on your description.
- Must be **smart** and **specific**.
  - Must clearly describe combination multiple techniques in a novel way.
- Must be **a unique idea** without a readily available solution online.
  - NOT: I will implement method XY in Python.
- **Cannot reuse work** from another course/project.
  - But can explore the same area.
- Only with our explicit approval.

Custom topic proposal

- Describe the intended features in a similar format as for the standard topics.
  - Bullet points, mark *3rd party code.*
  - Define both the **basic** and **extended** features.
  - Additional **custom** features can be added later during solving.
    - Same rules as for the standard topic apply.

- Send to aip-cs-ewi@tudelft.nl by **October 2**.
  - From a university e-mail with full name and student ID.

- We reserve right to accept, reject or propose changes.

Implementation

- Python or C++
- Platform not enforced (Windows, GNU Linux, Mac,…)
- You may use the code base from the **assignments**.

Implementation: 3rd party code

- *Green marked features*
  - You can use 3rd party code to implement this.
  - The **source of the code must be cited** in the code and in the readme.
    - Not necessary for libraries distributed separately (import numpy, #include <cv2.h>,..)

- Other features
  - You are expected to implement on your own.
  - You can still use libraries for low-level sub-steps.
  - **Example**: “Use RBF for interpolation”
    - **OK:** `np.linalg.inv(A)` to invert a matrix.
    - **NOK:** `xmath::interpolate(px,py,x,method='rbf')` to solve everything

Implementation: Code

- Source code should have **inline comments**.
  - The comments should be brief, but they should allow reader to understand what each piece of the code does.

- The **goal** is to verify that:
  - You **understand the algorithm**.
  - Your implementations is **reasonably** efficient, and the code does what the comment says.

- Code must compile and run on **Windows** or **Linux** (you can choose).

Implementation: Gitlab

- Use versioning system with the **provided Gitlab** repository.
  - Do not use public repositories (sharing code is not allowed).
- **Commit** changes frequently with meaningful **messages.**
- The instructors and TAs have access to your repository.

Gitlab access

- Web interface:

  https://gitlab.ewi.tudelft.nl/cgv/cs4365/student-repositories/2023-2024/cs436523[your_netid]

- Git:

  git clone git@gitlab.ewi.tudelft.nl:cgv/cs4365/student-repositories/2023-2024/cs436523[your_netid].git

Git Large File Storage

- In general, use LFS for medium sized files (5-100 MiB).
- In general, do not commit very large files (>100 MiB).
  - Provide a download link in the readme instead.
- Some more guidelines: https://docs.github.com/en/repositories/working-with-files/managing-large-files/about-large-files-on-github

Implementation: Readme.md

- Markdown document in the root of the submission with:
  - Pointers to corresponding **source files/methods/lines,** containing the implementation of **the algorithmic steps**.
  - **How to setup** the environment (what commands to run, what to install,…)
    - E.g.: `conda env create -f environment.yaml`
    - E.g.: `mkdir build & cd build & cmake ..`
  - **How to run** the program including sample arguments.
    - E.g.: `python my_code.py --scale 5 --algorithm fast input.jpg`
  - URL for the GitLab repository.

Implementation: Submission

- Submit a ZIP file to Brightspace (code + data + readme)
  - **Do not** upload:
    - Any **large files** (3rd party libraries, compiled binaries, testing/training data).
      - Provide a download link instead.
    - It is not needed to provide a detailed user documentation/manual.

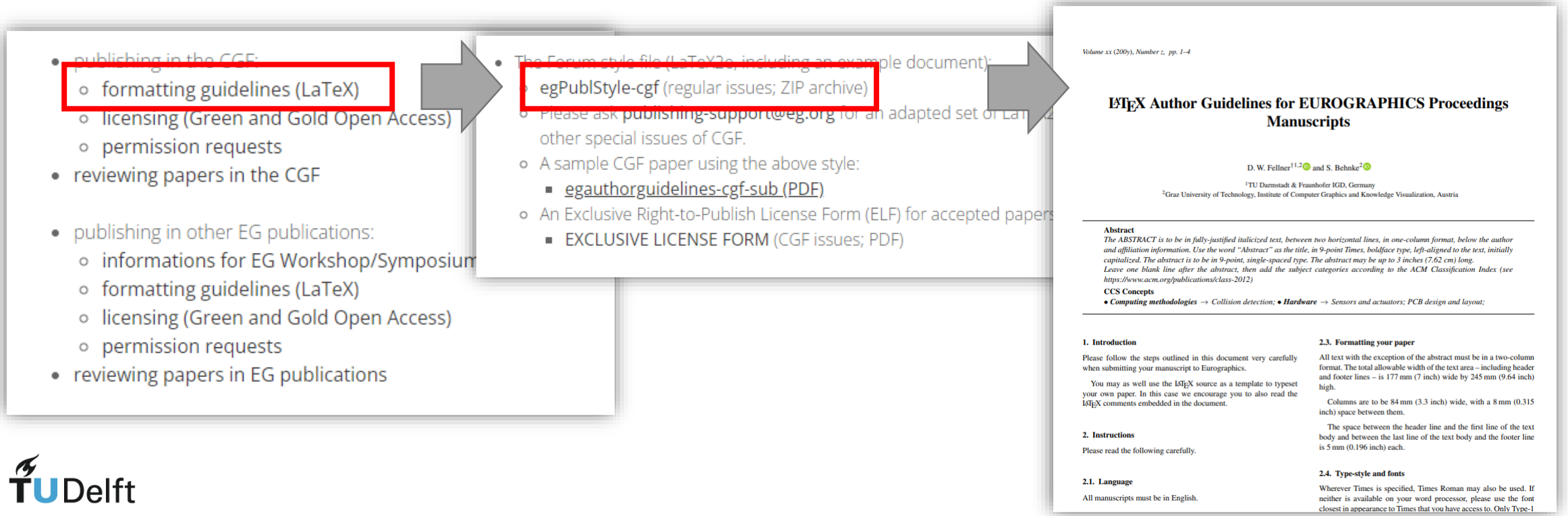
- (At least some) **test data** must be available:
  - **Small** files (<50 MB in total): ZIP them with the code.
  - **Large** files: Provide a download link in the readme.
    - The same for large libraries etc.

Implementation: DelftBlue

- You should all have access to **TU DelftBlue** computing cluster.
  - Provided you were enrolled on BS on **Sep 12**.
  - Mainly CPUs but also some limited GPU nodes.
  - https://www.tudelft.nl/dhpc
- You can run compute intensive work if you have no other option.
  - It is generally easier and more flexible to use custom environment.
  - There is a scheduler that deprioritizes work of heavy users.
- **Important instructions** are in BS->Content->Final project-> **Instructions for DelftBlue GPU cluster access**
- We do not directly support / teach how to use it.
  - Use documentation: https://doc.dhpc.tudelft.nl/delftblue/

Report

- **Very short visually focused** report:
  - Use the **Eurographics Template** (https://www.eg.org/wp/eurographics-publications/guidelines/)
  - We recommend LaTeX but feel free to use another text editor.



Report: Content

- **Title:** Specify which project you solve.
- **Author**: Your name and student ID.
- ~~**Abstract**: Not needed, skip.~~
- **Text**: Maximum ~200 words explaining your results and/or implemented features.
- **Figures**: 1-2 pages of **images** demonstrating your results and/or implemented features.

Video

- A short **voice-narrated** presentation (max. 4 minutes) made using **screen capture** (e.g., OBS).
  - Cuts and editing are allowed but it must be clear that the code indeed runs.

- **Content**:
  - **Code overview**: Structure, what is where… (~1 min)
  - **Run example**: Show inputs, run the code, show outputs.
  - Show additional results, features or other interesting artifacts.

- Presentations will be **pre-recorded** and uploaded to Brightspace as a .mp4 **video** (we recommend x264 codec).

Presentation rules (report or video)

- You need to show **intermediate results** of all steps for **at least one example input**.
  - Do this even if a step is not fully implemented, it is important for following.
  - Show the full pipeline for one input, then you can focus on specific steps with other inputs that highlight their features better.

- Steps that are not clearly demonstrated will **not be graded**.

Submission

- Submit everything together to **Brightspace** (zip file):
  - **Implementation** (code + readme + small data)
  - **Report** (pdf)
  - **Video** (mp4)

Final project: Timeline

- Standard topics available: ~~**September 25**~~ **-> September 20**
- Custom topic proposal: October 2 (optional)
- Submission deadline: **November 5**

Final project: Replacement

- An **individual** replacement project in the following quarter.

- TER Implementation Regulations Art 5, sub 5:
  - Only possible if the original Final project partial grade between 4 and 6.
  - The new Final project grade is limited to max 6.0.