The University of Melbourne
School of Computing and Information Systems
COMP90086 Computer Vision, 2024 Semester 2

Physical Reasoning Challenge

Project type:	Group (teams of 2)
Due:	7pm, 20 Oct 2023
Submission:	Source code and written report (as .pdf)
Marks:	The assignment will be marked out of 30 points, and will contribute 30% of your total mark.

A key application in computer vision is understanding physical relations between objects in the world. For example, self-driving cars must be able to use visual and other sensor data to reason about the location, speed, and trajectory of nearby objects in order to move safely on public streets. A robot working in a home or factory setting may need to use vision to solve everyday physics problems, such as how to pack items into boxes, how to place items onto shelves, or how to pick up and deposit items without breaking them.

In this project, you will develop an algorithm to solve a common visual physical reasoning task, which is predicting the stability of a stack of blocks from a single image. This task is challenging because the stacks of blocks have varying appearances, and they may appear against different backgrounds with different lighting and camera conditions. There are many possible ways to approach the task: for example, you might try to identify the individual blocks in the stack and their relative locations, or evaluate the 3d shape of the stack, or try to identify holistic features that distinguish between "stable" and "unstable" stacks.

Figure 1: AI-generated image from stablediffusionweb.com

Whatever methods you choose, you are expected to evaluate these methods using the provided data, to critically analyse the results, and to justify your design choices in your final report. Your evaluation should include error analysis, where you attempt to understand where your method works well and where it fails.

You are encouraged to use existing computer vision libraries your implementation. You may also use existing models or pretrained features as part of your implementation. However, your method should be your own; you may not simply submit an existing model for this problem.

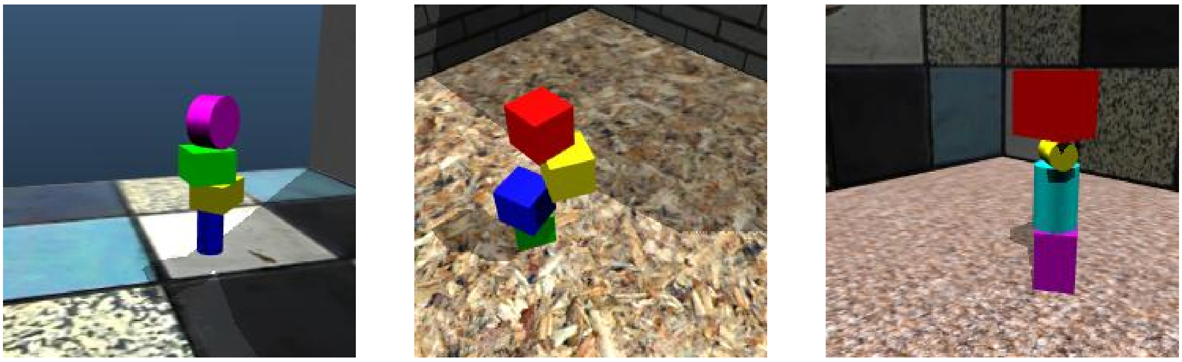

Figure 2: Examples of images from the ShapeStacks dataset [1]. The leftmost image shows a stable block stack, while the middle and rightmost images show unstable stacks.

Dataset

The dataset provided is a subset of the ShapeStacks dataset [1]. The images are synthetic scenes produced by a 3D rendering software. Each scene shows a vertical stack of blocks; the blocks have various colours and shapes (cubes, rectangular solids, spheres, cylinders) and the height of the stack varies from 2-6 blocks. Figure 2 shows examples of images from the dataset.

Your task is to build a model which can predict the stable height of the block stack, which is the number of blocks which have been placed appropriately such that they will not fall down. In a "stable" stack the stable height is identical to the total number of blocks. In an "unstable" stack the stable height is the height of the balanced portion of the stack; blocks above the stable height have been placed incorrectly and will fall down. Figure 3 shows diagrams of stacks which have a stable height of 3, which means that in each case the first 3 blocks have been placed correctly and are stable, but blocks above level 3 have been placed incorrectly and would fall down.

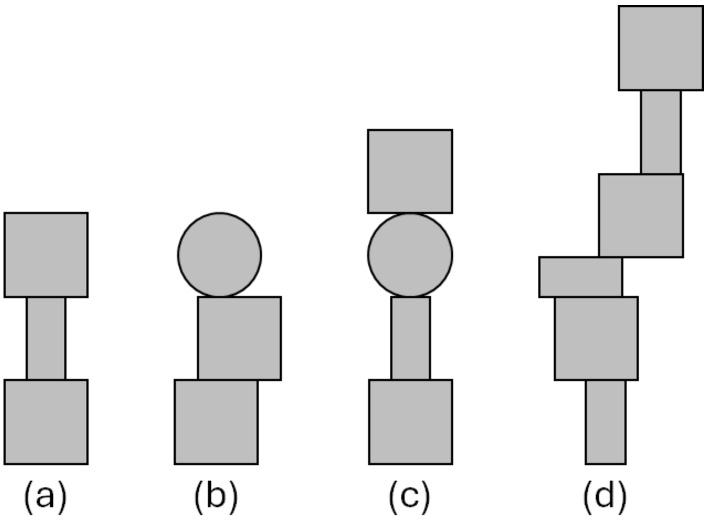

Figure 3: Diagrams of block stacks which have a stable height of 3. (a) and (b) are stable. In (c), the fourth block will fall down because a spherical block cannot support other blocks. In (d), the fourth and higher blocks would fall down because their centre of mass is not supported by the blocks below.

To build a stable stack, each block must be placed on a flat surface and the centre of mass of the block must be supported by the surface below. In unstable stacks, there is exactly one block placement that violates these rules. In all scenes, the bottom block of the stack is guaranteed to be stably placed, so the stable height of the stack cannot be less than 1.

You may use the provided training images however you wish to train your model. We have also provided some metadata for the training images, which you can use to help train or evaluate methods. You are allowed to use pretrained features, data augmentation, or supplemental images or data as part of your method. **However, your solution must not use any of following:**

- Additional images, supplemental data, or metadata from the ShapeStacks dataset which were not provided with this assignment
- Any models which were pretrained on the publicly-released ShapeStacks dataset

This assignment uses a custom train/test split of the dataset, so if you use the full public dataset or pretrained models, your method will be trained, in part, on the Kaggle test set. Any submitted method which involves training on the test set (or simply looking up the ground truth labels for the Kaggle test set) will be considered cheating and will receive 0 marks.

Scoring Predictions

You should submit your predictions for the test images on Kaggle. Your submissions for Kaggle should follow the same format as the `sample-solution.csv` file provided on the LMS. The file should include 2 columns:

- `id` = the image id (e.g., `'12345'`)
- `stable_height` = an integer indicating the stable height of the stack

The evaluation metric for this competition is **mean accuracy**: correct predictions will be scored as 1 and incorrect predictions as 0 and the total score is the average over the test set.

Kaggle

To join the competition on Kaggle and submit your results, you will need to register at `https://www.kaggle.com/`.

Please use the "Register with Google" option and use your @student.unimelb.edu.au email address to make an account. Please use *only* your group member student IDs as your team name (e.g., "1234&5678"). If your team name does not follow this format, we will not be able to identify your Kaggle submissions and you will receive 0 mark for the Kaggle portion of this assignment.

Once you have registered for Kaggle, you will be able to join the COMP90086 Final Project competition using the link under **Final Project: Code** in the Assignments tab on the Canvas LMS. After following that link, you will need to click the "Join Competition" button and agree to the competition rules.

Group Formation

You should complete this project in a group of 2. You are required to register your group membership on Canvas by completing the "Project Group Registration" survey under "Quizzes." You may modify your group membership at any time up until the survey due date, but after the survey closes we will consider the group membership final.

Submission

Submission will be made via the Canvas LMS. Please submit your code and written report separately under the **Final Project: Code** and the **Final Project: Report** links on Canvas.

Your **code** submission should include your model code, your test predictions (in Kaggle format), a readme file that explains how to run your code, and any additional files we would need to recreate your results. You should not include the provided train/test images in your code submission, but your readme file should explain where your code expects to find these images.

Your written **report** should be a .pdf that includes the description, analysis, and comparative assessment of the method(s) you developed to solve this problem. The report should follow the style of a short conference paper with **no more than four A4 pages** of content (excluding references, which can extend to a 5th page). The report should follow the style and format of an IEEE conference short paper. The **IEEE Conference Template** for Word, LaTeX, and Overleaf is available here: `https://www.ieee.org/conferences/publishing/templates.html`.

Your report should explain the design choices in your method and justify these based on your understanding of computer vision theory. You should explain the experimentation steps you followed to develop and improve on your basic method, and report your final evaluation result. Your method, experiments, and evaluation results should be explained in sufficient detail for readers to understand them without having to look at your code. You should include an error analysis which assesses where your method performs well and where it fails, provide an explanation of the errors based on your understanding of the method, and give suggestions for future improvements. Your report should include tables, graphs, figures, and/or images as appropriate to explain and illustrate your results.

Evaluation

Your submission will be marked on the follow grounds:

Component	Marks	Criteria
Report writing	5	Clarity of writing and report organisation; use of tables, figures, and/or images to illustrate and support results
Report method and justification	10	Correctness of method; motivation and justification of design choices based on computer vision theory
Report experimentation and evaluation	10	Quality of experimentation, evaluation, and error analysis; interpretation of results and experimental conclusions
Kaggle submission	3	Kaggle performance
Team contribution	2	Group self-assessment

The report is marked out of 25 marks, distributed between the writing, method and justification, and experimentation and evaluation as shown above.

In addition to the report marks, up to 3 marks will be given for performance on the Kaggle leaderboard. To obtain the full 3 marks, a team must make a Kaggle submission that performs reasonably above a simple baseline. 1-2 marks will be given for Kaggle submissions which perform at or only marginally above the baseline, and 0 marks will be given for submissions which perform at chance. Teams which do not submit results to Kaggle will receive 0 performance marks.

Up to 2 marks will be given for team contribution. Each group member will be asked to provide a self-assessment of their own and their teammate's contribution to the group project, and to mark themselves and their teammate out of 2 (2 = contributed strongly to the project, 1 = made a small contribution to the project, 0 = minimal or no contribution to the project). Your final team contribution mark will be based on the mark assigned to you by your teammate (and their team contribution mark will be based on the mark you assign to them).

Late submission

The submission mechanism will stay open for one week after the submission deadline. Late submissions will be penalised at 10% of the total possible mark per 24-hour period after the original deadline. Submissions will be closed 7 days (168 hours) after the published assignment deadline, and no further submissions will be accepted after this point.

Updates to the assignment specifications

If any changes or clarifications are made to the project specification, these will be posted on the LMS.

Academic misconduct

While it is acceptable to discuss the assignment with others in general terms, excessive collaboration with students outside of your group is considered collusion. Your submissions will be examined for originality and will invoke the University's Academic Misconduct policy (`http://`

academichonesty.unimelb.edu.au) where inappropriate levels of collaboration or plagiarism are deemed to have taken place.

Since Kaggle competition performance contributes to the final mark on this assignment, your submissions to Kaggle must be your own original work and must abide by the rules of the Kaggle competition. Submitting predictions to Kaggle which are not the results of your own models (or allowing another student to submit your model's predictions under their Kaggle account) will be considered a breach of the university's Academic Misconduct policy, as will any attempts to circumvent the rules of the Kaggle competition (for example, exceeding the competition's daily submission limit).

References

[1] O. Groth, F. B. Fuchs, I. Posner, and A. Vedaldi, "Shapestacks: Learning vision-based physical intuition for generalised object stacking," in Computer Vision – ECCV 2018, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018, pp. 724–739.