Teaching Data-driven Video Processing via Crowdsourced Data Collection

Max Reimann[1], Ole Wegen[1], Sebastian Pasewaldt[2], Amir Semmo[2], Jürgen Döllner[1] and Matthias Trapp[1]

[1]Hasso Plattner Institute, Faculty of Digital Engineering, University of Potsdam, Germany
[2]Digital Masterpieces GmbH, Potsdam, Germany

Abstract

This paper presents the concept and experience of teaching an undergraduate course on data-driven image and video processing. When designing visual effects that make use of Machine Learning (ML) models for image-based analysis or processing, the availability of training data typically represents a key limitation when it comes to feasibility and effect quality. The goal of our course is to enable students to implement new kinds of visual effects by acquiring training datasets via crowdsourcing that are used to train ML models as part of a video processing pipeline. First, we propose our course structure and best practices that are involved with crowdsourced data acquisitions. We then discuss the key insights we gathered from an exceptional undergraduate seminar project that tackles the challenging domain of video annotation and learning. In particular, we focus on how to practically develop annotation tools and collect high-quality datasets using Amazon Mechanical Turk (MTurk) in the budget- and time-constrained classroom environment. We observe that implementing the full acquisition and learning pipeline is entirely feasible for a seminar project, imparts hands-on problem solving skills, and promotes undergraduate research.

CCS Concepts
• ***Social and professional topics*** , . . . , ***Computing education;*** • ***Information systems*** , . . . , ***Crowdsourcing;***

1. Introduction

Motivation. Machine Learning is becoming a key technology for digital media, especially in image and video processing, and, coupled with the increase in computing power, has enabled a data-driven revolution in in the Computer Graphics (CG) domain. For example, deep learning-based super-sampling is part of many mainstream games [Nvi20], and the emerging field of neural rendering is seeing continued success in merging computer graphical concepts with neural-network based representations [TFT*20].

The usage of ML in graphics pipelines has also led to an increase in teaching of machine learning concepts and techniques as part of many CG curriculums. These courses and seminar projects typically focus either on teaching the theoretical and mathematical aspects of machine learning using small datasets for demonstration purposes or, in more hands-on courses, make use of pretrained models which in return base on available datasets. Being constrained to publicly available models and datasets presents a limitation to what kind of visual results can be created. Additionally, the robustness of the solution can suffer when the targeted application domain (e.g., dimly lit imagery) is not represented in the training data. Building a custom dataset and learning pipeline, on the other hand, lends the freedom of defining the desired model outputs first and then concluding the requirements for dataset and model.

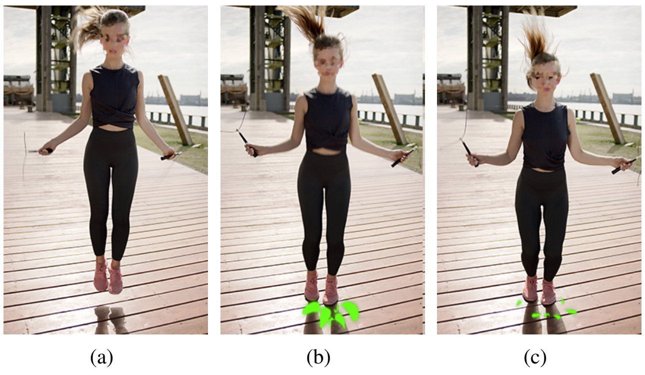


Figure 1: *Exemplary frames from the developed contact-visualization video effect: (a) person not touching the ground, (b) person initially touching the ground yield a strong splash effect, (c) a subsequent frame of person touched the ground with a weaker splash effect.*

To validate this concept, we guided a seminar project to implement such an ML-based graphical effect (Fig. 1) for that no pretrained model or suitable dataset exists. Concretely, the goal of the seminar project was to detect and stylize human contact points with their environment in videos. The scope of this topic covers the computer vision domain of action detection and localization in videos, as well as the computer graphical domain of video post-processing with human-centric effects. While some related work on detecting feet contact exists [ZYC*20, RGH*20], these methods only consider static cameras, a small variety of movements, and data which was captured in controlled or synthetic environments. In contrast thereto, the research goal for this seminar project was to train a neural network to perform such detections on User-Generated Content (UGC), e.g., videos in the wild as found on streaming platforms such as YouTube, which contain a wide variety of scenes and camera motion not found in a static setup and mandated to create a custom dataset.

Challenges of Training Data Generation. However, how to collect training data is a rarely practically taught aspect of the deep learning pipeline. The lack of practical teaching content regarding this topic in CG courses is not surprising, as collecting enough data to train a deep learning model is often considered too complicated or expensive for a classroom project. While for some tasks such as fine-tuning a classifier to a new class, manual data annotation performed by students can be sufficient, for more challenging computer vision domains such as training on video data, manually annotating enough data is infeasible for a group of students. To create larger datasets, typically, specialized companies are tasked who employ experienced annotators. These companies commonly charge per image and have different pricing depending on the annotation complexity, e.g., annotating 50 000 images with segmentation masks using Google's annotation service [Goo21] will cost $43 500 (USD) at the lowest quality level (one annotator per image), which is clearly not affordable for a research or seminar project. Alternatively, crowdsourcing the data annotation on a microtask marketplace such as MTurk [Cro12] offers the benefit of an order of a magnitude cheaper cost of labor, that can make it affordable even in the context of a CG classroom. However, there are some unique challenges to overcome: (1) designing an intuitive and efficient annotation tool, (2) preventing incorrect results, as well as (3) encouraging consistent and high quality.

Approach and Contributions. With respect to these challenges, this paper presents the approach of teaching hands-on knowledge in data-driven solving of CG problems by instructing a seminar project to design a video annotation tool, collect a dataset using crowdsourcing and training an ML-model on the acquired data, using an iterative feedback cycle to improve the different aspects of the process along the way. The presented approach helps students to gain hands-on knowledge for solving computer graphical tasks using custom ML solutions, that can enable new applications and promotes undergraduate and graduate research [AAF16]. To summarize, this paper makes the following contributions to the reader:

1. We describe a reproducible and comprehensible workflow with best practices to create customized MTurk workflows in the context of a data-driven CG seminar.
2. We show the feasibility of our seminar concept on the example of a project for contact detection, the proposed concept can however be applied to any ML-focused CG seminar
3. We discuss and evaluate our approach and its design decisions based on the results of a qualitative and quantitative evaluation.

2. Background & Related Work

Background. Using MTurk, annotation microtasks—also called Human Intelligence Tasks (HITs)—are submitted to the marketplace and completed by thousands of anonymous non-experts for a fee, which is chosen by the annotation requester. While crowdsourcing platforms such as MTurk offer templates for certain standard tasks, solving new research problems generally also requires the analysis, design, and implementation of specialized data annotation tools. The tool design is challenging, as it must balance the following aspects:

Clarity: The task definition and tool usage have to be made as clear as possible, as most users on MTurk are not trained in CG or annotating datasets. Furthermore, as there is considerable pressure of profitability for workers to complete tasks as quickly as possible, they often only skim read instructions and thus create faulty annotations.

Accuracy: The annotation tool has to enable annotators to consistently achieve a high level of accuracy when annotating images.

Efficiency: The tool must enable the annotator to complete a task in the minimal amount of time possible, as the available budget for the project is limited and a high number of annotated images is necessary to train a computer vision model to high accuracy.

Related Work. Collecting crowdsourced annotations has been previously studied in the context of many different fields and applications. Sorokin and Forsyth were one of the first to describe the usage of MTurk as a generic tool for crowdsourcing data annotation [SF08]. Kovashka *et al.* [KRFFG16] provide a comprehensive survey of research works using crowdsourcing for computer vision, providing a good overview of domain-specific approaches. A major focus in studies on the crowdsourcing marketplace is the issue of evaluating and improving the result quality [WP10]. Rashtchian *et al.* [RYHH10] find that for annotating images with free-form labels, the use of a qualification test provides the highest accuracy amongst available quality control methods—we therefore choose to implement qualification tasks in our workflow.

Kovashka *et al.* [KRFFG16] note that using crowdsourcing for large-scale video annotation remains challenging due to the size of the data and the difficulty of designing efficient interfaces, and often mandate custom interfaces. We confirm this statement, as available video annotation tools, e.g., *LabelMe* [RTMF08] or *VATIC* [VPR13], could not efficiently be used for fine granular, frame-by-frame annotations of video events, as is required for contact detection. We thus decided to build our annotation tool from scratch.

While crowdsourcing has been used in several capacities in the classroom, mostly focused on creating educational content, distributed feedback and grading [JSB18], to the best of our knowledge, we are the first to propose combining collection of an deep learning-scale dataset through crowdsourcing and subsequent model learning and application in a seminar project.

3. Course Concept

3.1. Academic Environment

The presented concept was implemented as part of a master's seminar on image and video processing at the Computer Graphics Systems (CGS) Group of the Hasso Plattner Institute at the University of Potsdam. The course was designed for 6 European Credit

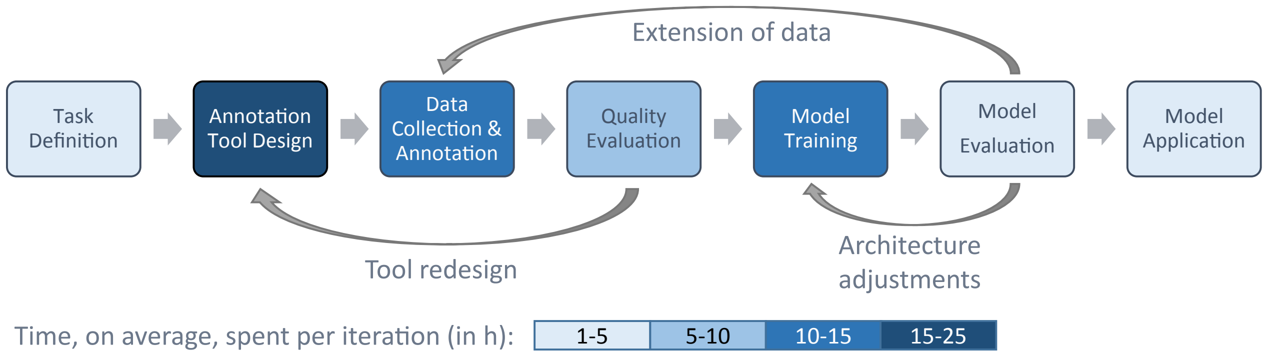


Figure 2: *Overview of the project implementation steps in a data-driven CG seminar. Development progresses in an iterative way: after designing an initial annotation tool, a test batch of annotations is collected, and if necessary, tool design and quality assurance are improved before launching a new annotation batch. Similarly, the model is trained on the available data and depending the model performance, new data is collected, or the architecture or training loss functions are adjusted. The supervisor is involved in all steps, gives suggestions and feedback on design decisions, and suggests when to advance to the next stage. In the seminar, three iterations for tool redesign (batches 1-3 in Table 1) and one training data extension (batch 4), as well as one major architecture change were made.*

Transfer System (ECTS) points, relating to 180 h of study time, i.e., 10 h to 12 h per week and had a student-advisor ratio of 2:1. Apart from the basic course through requirements of having successfully participated in an image processing or CG lecture, for this project it was also required to have a basic theoretical understanding of ML, and training deep learning models in particular. Diving deep into an open research question and building a full deep learning pipeline from data acquisition to training is a daunting task for students without appropriate pre-knowledge. Therefore, in practice, it is advisable to select students that already gained hands-on experience in the data domain. In this case, the participating student had already gained a good working knowledge of relevant computer vision techniques from previous seminars.

3.2. Course Design

The seminar design follows the concept of Trapp *et al.* [TPD*18] that comprises three phases, which are briefly described as follows.

A. Course Introduction: The student is introduced to the domain by providing him with relevant related work in this area and is tasked to read and summarize these works as well as search and keep an up-to date bibliography of related and relevant work during the seminar. The first meeting of the supervisor with the student serves to define requirements for the project, and a document is set up in which the must-haves, should-haves and nice-to-haves are defined, which in return can be used to work out a roadmap and later on helps to assess the progress of the student.

B. Project Implementation: The main work in the seminar naturally lies in the various aspects of dataset collection and model training. After on-boarding, the student together with the supervisor first defines the data requirements and annotation task (Sec. 4.1), and then starts dataset collection (Sec. 4.2) and implementation of the annotation tool (Sec. 4.3), while ensuring the quality of collected data (Sec. 4.4), and, finally, works on model learning (Sec. 4.5) and visual effects engineering. The project implementation follows an agile process, as shown in Fig. 2, during which the supervisor is steadily kept in the loop. For each stage, the supervisor and student discuss their respective ideas and work out a detailed task specification. The student then implements a prototype for this stage while iteratively receiving feedback from the supervisor, who advises the student on when to move forward (e.g., with data collection or training). Furthermore, the supervisor helps to find and field appropriate data(sets) to annotate, suggests directions to take during ML-model implementation and training, and keeps an overview of the timeline and progress. During project implementation, the student freely chooses the languages and frameworks that he is most comfortable with and is advised to take the most straightforward and easy-to-implement approaches first - such as designing a minimal annotation tool as opposed to making use of sophisticated web-frameworks. This reduces the technical risk and uncertainty for the student.

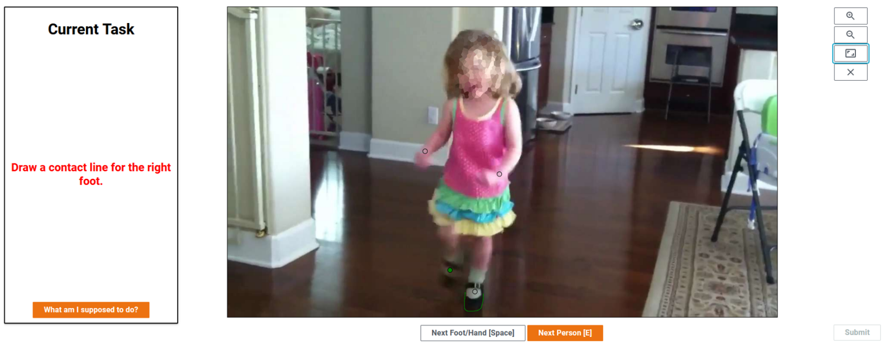
C. Presentation & Grading: The student also presents his progress in an 8 min to 10 min long intermediate presentation in front of the class and instructors, in which the student shows his annotation and model training concept, as well as the first prototype of the annotation tool, and receives feedback from his classmates and other instructors. After the seminar is complete, the student presents his results in a 20 min long talk plus 10 min open discussion. The evaluation and grading of the project follows the best practices of [TPD*18].

4. Dataset Creation

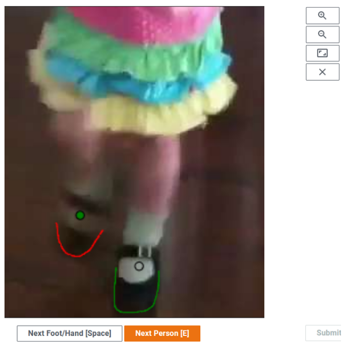
This section describes the major aspects of the crowdsourced data collection during the course's project implementation.

4.1. Data Requirements and Labeling Task Definition

After completing a review of related work on the domain-relevant state-of-the-art methods in the introduction phase, the capabilities, limitations, and training data of these methods should be documented in terms of how well they match with the requirements of


Current Task
Draw a contact line for the right foot.
What am I supposed to do?
Next Foot/Hand [Space]
Next Person [E]
Submit

(a)



(b)

Figure 3: *The first iteration of our video-frame annotation user interface (a), with a zoomed-in image (b). Circles mark the detected keypoints, the currently active keypoint is marked in green. The current task (i.e., which body part to annotate) is shown on the left as well as a link to the annotation tutorial. When drawing annotation lines, each body part is assigned a different line color. The "Next Foot/Hand" button cycles through the keypoints, while the "Next Person" button is pressed if no further keypoints are found to be in contact for the current person. Using the tools on the upper right, the user can zoom in and out (alternatively using the mouse wheel), reset the zoom scale, and reset the last annotation. Keyboard shortcuts are available.*

the seminar project. The next step then is to specify the model input and output modalities. Staying in line with related work, where possible, will help to reduce uncertainty about the feasibility of training a model using the collected data. Detailed training data requirements should be specified such as the input resolution, content class distribution, video length and Frames-per-Second (FPS). After deciding what kind of output modalities a model should learn, the ground truth label format can be determined. The annotation task that generates this ground truth has to be very precisely defined and structured in such a way as to optimize label accuracy while minimizing worker time and effort at the same time [KRFFG16]. Any flawed design decision that is not caught on early in the process can potentially lead to very costly workflows and even unusable data. Therefore, common annotation definitions and protocols (e.g., from related datasets) should be reused. This eases the comparison with published methods and enables re-training of published models on the newly created dataset. The completion of the data and task definition phase (Fig. 2) marks the beginning of tool implementation and dataset collection.

Defining Contact Annotations. After reviewing relevant related work, we deviate from previous work on contact detection, which solely predicts a contact-state for a keypoint, to rather generate free-form annotations of the visible contact line. More formally, per-person in the image and for each hand or foot respectively, the visible border where the body part is in contact with the scene, is to be traced along with a line (Fig. 3(b)). The annotations are stored as pixel maps per person. The rationale behind annotating a precise contact line in contrast to just predicting the contact state is, that it can improve the capability of a network to learn body part placement and orientation in the environment, as well as enable more accurate placement of graphical effects. The collected videos have a length of 4 sec each and are sampled at 10 FPS.

4.2. Data Collection and Pre-processing

To collect source material (e.g., videos) for annotation, existing large-scale datasets can often be re-used, which significantly reduces data-scraping efforts. Specialized search engines such as *Google Dataset Search* help with finding appropriate source datasets. However, these datasets might not comprise the quantity, quality, or distribution of classes required for the task at hand. For example, in the computer vision domain, the number of image datasets is still considerably larger than those focusing on video, largely because of the high costs associated with creating video data annotations. Furthermore, datasets containing "in the wild" data sourced from online platforms such as YouTube are quite rare compared to synthetic data or datasets created in controlled environments. Collecting additional custom data (from the web) can thus be necessary to create or complement the dataset. While creating a video dataset with individually annotated frames is very labor-intensive in the annotation phase, the collection of the source data can be completed fairly quickly through targeted keyword search and extraction of large number of frames per video.

Collecting Videos for Contact Annotation. First, large-scale video action-recognition datasets such as AVA [GSR*17] were fielded to include a large variety of different scenes and classes of activities, that appear in user-generated videos. Specifically, videos were selected to have (1) a variety of human motion (e.g., dancing, walking, jumping), (2) different numbers of persons, (3) varied surroundings, and (4) static as well as dynamic cameras. Afterward, videos for categories that were underrepresented or considered very relevant and challenging were manually collected from sources such as YouTube. This was done by searching for larger video compilations of specific categories, such as parkour or break-dancing, and then skimming the video and setting time markers at the beginning of relevant video content. To process these time markers, a data preprocessing pipeline was built that encompassed (1) downloading the video, (2) trimming it into 4sec slices starting at each time marker, (3) predicting 2D keypoints to mark foot and hand positions, (4) filtering these keypoints, and finally (5) generating a Comma-separated Values (CSV) file for further usage in MTurk. The complete dataset was collected, in iterative steps, as shown in

Fig. 2, starting with smaller batches in between tool design iterations to a large data extension after the first model training.

4.3. Design of Annotation Tool

Building annotation tools for a crowdsourcing platform can be a time-intensive task due to the required expenditure for setting up data pipelines and developing the front end in a way to accommodate both, accuracy and efficiency. MTurk provides User Interface (UI) templates for certain categories of common tasks, e.g., classification and segmentation. However, less common tasks such as video annotation will require the implementation of a custom annotation tool, which can be embedded as an `IFrame` into the MTurk website. Most commonly, such an annotation tool is implemented by developing a HTML/JavaScript (JS) web-page and serving media from a dedicated server. Our annotation tool, as shown in Fig. 3, was designed to balance clarity, accuracy and efficiency, as described in the problem statement (Sec. 1). The respective design decisions are described in the following.

Labeling Instructions. To avoid inaccurate annotations originated from misunderstood task specifications, it is vital to provide detailed, but easy-to-understand instructions to workers. In the tool UI, the current task is highlighted on the left side, and a tutorial is shown when clicking on the "What am I supposed to do" button (Fig. 3), which includes annotation instructions and several scenarios of how and when to annotate which images. While the annotation tool produced accurate results when testing with coworkers and other students, in an annotation test run on MTurk, the error rate was extremely high, resulting in only 35 % of the results correctly annotated (batch 1 in Table 1). Typical errors ranged from spatial inaccuracies to totally ignoring the assignment instructions (Fig. 5). We speculate that many workers did not or only superficially read the task instructions and annotated every hand and feet irrespective of the contact state. Reasons might include time pressure and misunderstanding the task as a generic hand/feet labeling task, which are quite common on the marketplace. The tool was redesigned - in its first redesign iteration (Fig. 2) - to include a step-by-step tutorial (Fig. 4) that presented a quiz at the end to test if candidates had read and understood the instructions and would be able to confidently annotate the results. This resulted in an increase in the ratio of correct results to 64 % (batch 2 in Table 1).

Accuracy Guides. Depending on the task type, different levels of accuracy are necessary or can be achieved. When dealing with image annotations, the worker should receive tool guidance to help accurately place labels or draw annotations, ideally by using known image editing metaphors. To help workers more accurately annotate images, we implemented buttons and shortcuts for image zooming (using the mouse-wheel), panning, scale reset, undo & redo. These helped annotators to draw accurate pixel-precise lines by zooming in on the contact boundaries before drawing a contact line (Fig. 3).

Increasing Tool Efficiency. In a budget-constrained setting, optimizing the time spent per annotation is of high importance. The main factor to increase throughput is to design the tool for ease of use and giving visual guides to annotators, where possible, to focus attention on the relevant parts. When annotating multiple frames of a video, presenting subsequent frames increases the annotation speed versus randomly ordered frames [VPR13], as the worker benefits from memorizing the ongoing motion.

Furthermore, as Vondrick *et al.* [VPR13] note, minimizing interruptions and available interaction choices can significantly reduce user anxiety and increase efficiency. Therefore we seek to restrain the number of possible actions a user can complete. In our experiments, the time to move the mouse to the human contact position, zoom in, select the correct label and start annotating took nearly as long as the annotation itself. We therefore opted to pre-generate pose keypoints, which exclusively serve as a visual guide to the body part. These guides help workers to annotate and trigger the auto-selection of the appropriate body-part label. We generate keypoints using OpenPose [CHS*18], since it is the state-of-the-art method for 2D keypoint detection. The annotator cycles through the different keypoints by clicking on "Next Foot/Hand" or hitting space. To reduce time spent on zooming in on contact-positions, the user selects the zoom level when annotating the first body part of an image, which is then applied as an auto zoom-in for the subsequent keypoints for this image, and is reset after every image. Therefore, annotators do not need to interrupt their current drawing workflow to pan and zoom in on every new keypoint, thereby reducing annotation time by approximately 30 %.

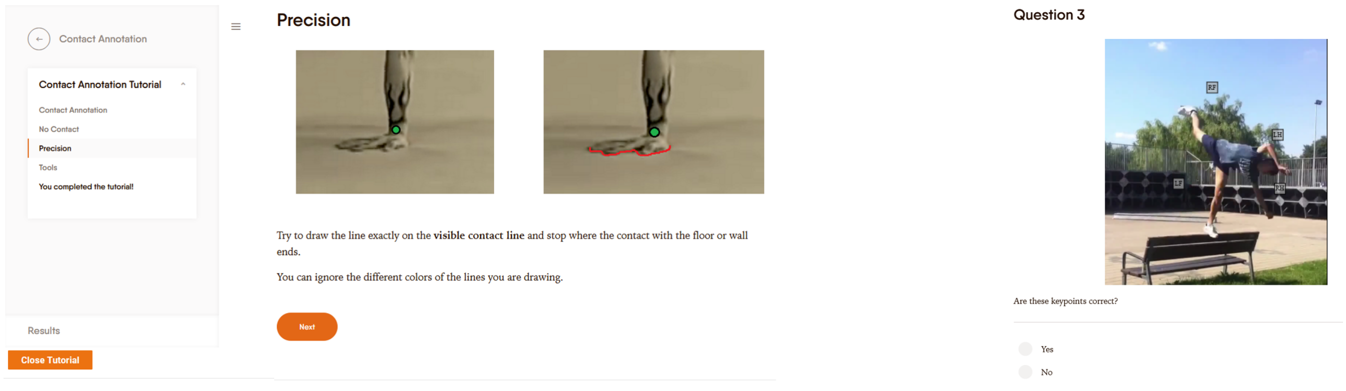
Keypoint Correction. In some cases the pose-prediction did not correctly generate keypoints, which led to annotators being unable to annotate some body parts or labeling incorrect positions, e.g., in the case that pose-prediction switched left and right-foot keypoints. To remedy this, a keypoint correction step was added before the annotation step, in which annotators had to decide if the keypoints that had been predicted were correct Fig. 6. While the price per HIT had to be upgraded from $0.18 to $0.30 to accommodate for the increased time spent, the result quality was significantly increased.

Reducing Annotator Frustration. Annotation of large volumes of images is a very repetitive task and can easily frustrate annotators quickly. On the other hand, keeping accurate workers on the task as long as possible is beneficial to profit from their learned annotation efficiency. We observe, that decreasing the number of interactions per HIT and increasing guides helped to keep workers motivated for longer. Furthermore, monetary incentives and direct communication to our best workers, as described in Sec. 4.4, were implemented to increase worker retention.

4.4. Data Evaluation and Quality Assurance

Ensuring the quality of annotations is one of the most important aspects when crowdsourcing the creation of annotations to unskilled workers. According to Sorokin and Forsyth, there are three strategies for assuring quality [SF08]:

1. Building a gold standard, which is a collection of images annotated by the researchers themselves, and injecting it into the running annotation batch for every worker. Using an error metric or manual inspection, one can quickly detect workers who are delivering subpar results. However, specifying an error metric might not be possible and manual inspection too time-consuming.
2. Collecting multiple annotations for each input. Here, some sort of consensus protocol must be implemented, which can include



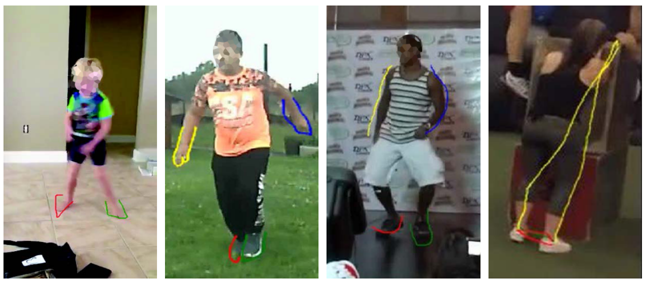
Try to draw the line exactly on the **visible contact line** and stop where the contact with the floor or wall ends.

You can ignore the different colors of the lines you are drawing.

Are these keypoints correct?

(a) (b)

Figure 4: *Annotation tutorial (a) gives a step-by-step explanation of the annotation process with examples for correct and incorrect annotations. The following quiz (b) tests the workers understanding of the tool and task by asking if generated keypoints are correct (if not, workers have to manually annotate at the correct body parts during annotation), and if a shown annotation is correct.*



(a) (b) (c) (d)

Figure 5: *Examples of wrong annotations, often seen using the first version of the annotation tool: (a) too low spatial accuracy, (b) marking of hand/feet not in contact, (c) wrong/misinterpreted arm annotation, and (d) did not read task description.*

additional information such as worker history and label quality estimates [WP10]. However, using multiple annotators per image will at least halve the number of annotated images, which might be infeasible due to budget constraints, or is not possible because differing annotations could still be correct.

3. Using a grading task. Workers grade the results of others, thereby creating a ranking in the result quality. Such a grading task can be implemented at a much lower cost than the original annotation task, however is also subjective, especially for free-form annotations.

Qualification Tasks. In the seminar project, a variation of the gold standard (1) and grading task (3) was employed, which is supported by MTurk as a "qualification task". Before being admitted to the actual annotation tasks, annotators had to fulfill a standardized task and complete at least 95 % of the annotations correctly. In contrast to the verification methods found in literature, the verification of qualification tasks was done manually, as no simple way of algorithmically verifying free-form annotations, such as the created contact mask annotations, exists. While the quality evaluation phase (Fig. 2), which included manual worker verification and approval, proved to be a time-intensive task, many workers who did not fulfill basic requirements such as annotating all images or completing the initial test-round were filtered out before reaching the manual result verification stage. Workers were incentivized to meticulously complete the tutorial and qualification task by giving a $1 bonus to every worker who successfully qualified.

Incentives. After recruiting capable workers through qualification tasks, consistency of annotation quality can be promoted by incentivizing the most accurate annotators to keep on working. After each batch of annotations, the results of each worker were sampled. Workers with very accurate results and consistent work progress were awarded bonuses to incentivize further work while keeping up high accuracy standards. Furthermore, the MTurk Application Programming Interface (API) also allows sending individual messages to workers, which was used to attach feedback to high-volume annotators on what aspects to improve. The introduction of these incentives led to approximately 3 % of the workers creating 80 % percent of the task annotations (Fig. 7).

4.5. Model Training Phase

Starting to train models already with only partially collected data can help to determine for which cases models perform well and which cases need more data. In a data extension step (Fig. 2), the images are then collected according to the required class distribution. We found that retraining the network after a batch of around 2000 HITs was completed on MTurk presented an efficient way of judging the effect of newly annotated images.

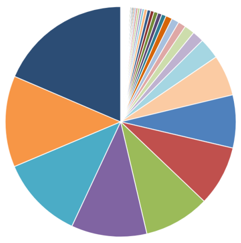
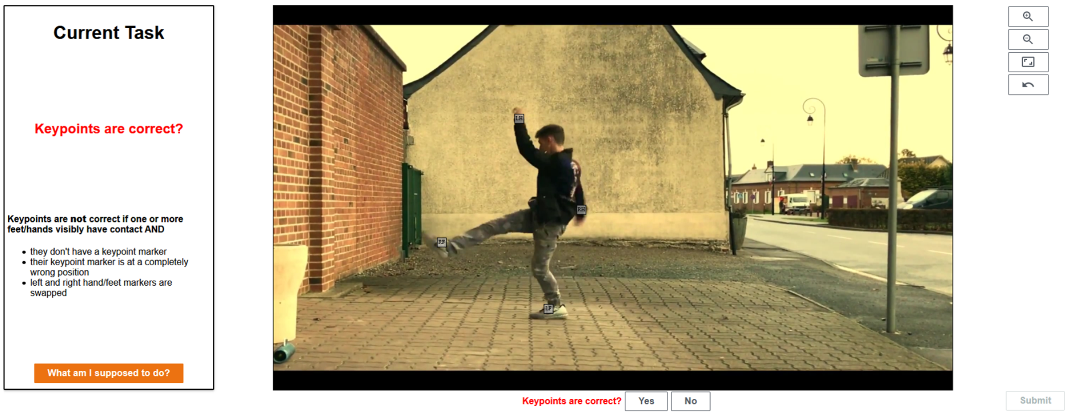


Figure 7: *Percentage of tasks completed per annotator. While there were over 290 annotators in total, only a small number of dedicated workers completed the large majority of the HITs.*


Current Task

Keypoints are correct?

Keypoints are not correct if one or more feet/hands visibly have contact AND

- they don't have a keypoint marker
- their keypoint marker is at a completely wrong position
- left and right hand/feet markers are swapped

Figure 6: *A keypoint correction step before every annotation step to handle incorrect keypoint predictions. If the worker selects that keypoints in the shown image are not correct, the subsequent body parts have to be then annotated without any keypoint guides.*

5. Evaluation, Results, and Discussion

5.1. Resulting Dataset and Pricing

In total, 46 000 instances (persons), collected from 760 videos, were annotated, of which 36 000 annotated instances were usable. As Table 1 shows, the introduction of a quiz (batch 2) and qualification tasks (batch 3) led to a significant increase in usable results, and a further minor was seen in batch 4, after workers had received individual feedback on their annotation results. In total, the creation of the dataset (including test batches) cost $1650.

Pricing in MTurk is set per HIT and there is no way for workers to know in advance what their pay per hour will be. Many workers will join at any price level, but might quickly quit if they feel underpaid. Keeping efficient workers motivated can be achieved through fair pay and occasional incentives, as described in Sec. 4.4. To determine a fair pricing point, we set up 20 HITs each at three different price points, corresponding to wages of $3/h, $6/h, and $8/h. While we observed no significant difference between the latter two tasks, workers for the $3/h task were around 50 % less likely to accept another HIT of this type and spent overall less time reading the instructions. Therefore, the approximate $6/h pay was used in all consequent annotation batches.

5.2. Course Evaluation

Feedback on the course was collected using the course grading form described in [TPD*18]. Overall, the course was well received in terms of time expenditure, course enjoyment and motivation for further research and received an excellent grade in terms of gained knowledge. Naturally, because of the small sample size, these quantitative results are merely an indication. The time expenditure for the student was in line with the 180 h total study time pertaining to the 6 ECTS credit points, of which, roughly 60 h were spent on tool design, 40 h on data annotation and evaluation, and 40 h on model training and application. The workload for the supervisor was higher than for other CG courses, since the supervisor took a more active role during the ML-training process, as the student had only little experience in this regard.

Batch	HITs	Price-per-HIT	Quiz	Qual. Task	Useable
1	886	$0.18	no	no	35 %
2	498	$0.18	yes	no	64 %
3	1,659	$0.30	yes	yes	82 %
4	1,565	$0.30	yes	yes	89 %

Table 1: *Annotation Batches. Each HIT consisted of 10 persons in total to annotate. In batch 3 & 4, the extended UI with an extra keypoint correction step (Sec. 4.3) was used, therefore the price per HIT was increased. Batch 4 was completed after high-volume workers had received individual feedback.*

Further, the student was asked to give qualitative feedback by answering several questions pertaining to the course, especially focused on the crowdsourcing and ML part, which is presented in the following. The question *"what are some of the key learnings you took from this project"* revealed that the student learned that a precise and clear definition of annotation tasks is crucial and making the annotation tool/task as unambiguous as possible as well as using qualification tasks was important to obtain good training data. Further, the student experienced that training and fine-tuning a ML model is a tedious and lengthy process with a lot of trial-and-error and felt that having a supervisor who is experienced in this field was important to progress in a timely manner. Overall, knowledge about ML training and improved skills in Python and JS were acquired. The student agreed that *"gaining hands-on knowledge on data acquisition is a valuable part of a CG/ML curriculum"* and stated that the quality of the data influences the results so profoundly that it is important to know, how to acquire such data for training specific tasks.

The student was asked about the strengths/weaknesses of approaching an image & video processing seminar in a data-driven manner. As a strength, the student viewed the holistic approach and therefore deeper insights into video processing with ML, and also not being bound to existing models. As a weakness, he stated that achieving visually appealing stylizations is usually straightforward using pre-trained models, while the data-driven approach bears the risk of possibly not achieving a satisfying result at all. Also, waiting on long training runs and the arduous process of source data acquisition (fielding and collecting videos) that did not impart much new knowledge about image & video processing were stated as a downside. All in all, the student felt *"more confident in tackling a CG/video processing problem using ML in the future"*, especially regarding the data acquisition aspect. Regarding training deep models, the student stated that he gained a lot of hands-on experience, and wants to further deepen his knowledge in this field.

5.3. Discussion and Future Work

Overall, the concept of using crowdsourcing for training data acquisition during a CG course received positive feedback. As this was the first offering of the course, there were challenges along the way. Particularly in the beginning, the participating student felt that the next steps and exact roadmap were unclear, as the project proceeded along an iterative cycle of data collection, annotation tool redesign and model training. This will be improved by a more detailed roadmap, containing quantifiable milestone goals to reduce the student's uncertainty about his progress in the seminar. Extending the participant size to form a group project of 2 to 3 persons would enable faster iteration cycles and speed up source data collection, leaving more time for model training and implementing high-quality visual effects. For an optimal learning experience, each student should participate and experience each step of the task pipeline (Fig. 2). How to efficiently distribute workloads while providing hands-on knowledge about every step to each student is thus an interesting subject for future work.

Next to equipping students for further research in the CG and ML domains, the course concept also provides tangible benefits for instructors, as data and insights gained in this project could be directly used for active CG research. A downside of this approach is that per project approx. \$1000 to \$2000 need to be spent for MTurk workers, which limits the number of projects being able to follow this approach in one course. Scaling up the number of projects could be achieved in future courses by making use of synthetic data generation to pre-train ML models, and then using a reduced set of crowdsourced annotations to fine-tune the model to real-world imagery.

6. Conclusions

This paper presents the concept for a data-driven project seminar in the CG domain, using crowdsourced data-acquisition, model training, and visual effects design implemented in a holistic pipeline. For this, best practices for custom data acquisition workflows in MTurk for budget constrained environments such as a seminar classroom are provided. The paper presents and evaluates these practices on the concrete example of an excellent student project in the domain of video processing. The proposed concept generalizes to any ML-focused teaching and learning environment, delivering a straightforward approach to promote undergraduate and graduate research.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable feedback. This work was funded by the German Federal Ministry of Education and Research (BMBF) through grant 01IS15041 ("mdViPro") and 01IS19006 ("KI-Labor ITSE").

References

[AAF16] ANDERSON E. F., ADZHIEV V., FRYAZINOV O.: Aiming high: Undergraduate research projects in computer graphics and animation. In *Proc. Eurographics Education Papers* (2016), Eurographics Association, pp. 17–24. 2

[CHS*18] CAO Z., HIDALGO G., SIMON T., WEI S.-E., SHEIKH Y.: OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. In *Proc. IEEE CVPR* (2018), pp. 1302–1310. 5

[Cro12] CROWSTON K.: Amazon mechanical turk: A research tool for organizations and information systems scholars. In *Proc. IFIP WG 8.2 Working Conference* (Berlin, Heidelberg, 2012), Springer Berlin Heidelberg, pp. 210–221. 2

[Goo21] GOOGLE: Data Labeling Service, Google Cloud, 2021. 2

[GSR*17] GU C., SUN C., ROSS D. A., VONDRICK C., PANTOFARU C., LI Y., VIJAYANARASIMHAN S., TODERICI G., RICCO S., SUKTHANKAR R., SCHMID C., MALIK J.: AVA: A Video Dataset of Spatio-temporally Localized Atomic Visual Actions. In *Proc. IEEE CVPR* (2017), pp. 6047–6056. 4

[JSB18] JIANG Y., SCHLAGWEIN D., BENATALLAH B.: A review on crowdsourcing for education: State of the art of literature and practice. In *Proc. PACIS* (2018). 2

[KRFFG16] KOVASHKA A., RUSSAKOVSKY O., FEI-FEI L., GRAUMAN K.: Crowdsourcing in Computer Vision. *Foundations and Trends in Computer Graphics and Vision 10*, 2 (2016), 103–175. 2, 4

[Nvi20] NVIDIA: NVIDIA DLSS 2.0: A Big Leap In AI Rendering, 2020. 1

[RGH*20] REMPE D., GUIBAS L. J., HERTZMANN A., RUSSELL B., VILLEGAS R., YANG J.: Contact and human dynamics from monocular video. In *European Conference on Computer Vision (ECCV)* (2020). 2

[RTMF08] RUSSELL B. C., TORRALBA A., MURPHY K. P., FREEMAN W. T.: Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision 77*, 1-3 (2008), 157–173. 2

[RYHH10] RASHTCHIAN C., YOUNG P., HODOSH M., HOCKENMAIER J.: Collecting image annotations using amazon's mechanical turk. In *Proc. NAACL HLT Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk* (2010), pp. 139–147. 2

[SF08] SOROKIN A., FORSYTH D.: Utility data annotation with Amazon Mechanical Turk. In *Proc. CVPR Workshops* (2008). 2, 5

[TFT*20] TEWARI A., FRIED O., THIES J., SITZMANN V., LOMBARDI S., SUNKAVALLI K., MARTIN-BRUALLA R., SIMON T., SARAGIH J., NIESSNER M., PANDEY R., FANELLO S., WETZSTEIN G., ZHU J.-Y., THEOBALT C., AGRAWALA M., SHECHTMAN E., GOLDMAN D. B., ZOLLHÖFER M.: State of the Art on Neural Rendering. *Computer Graphics Forum 39*, 2 (2020), 701–727. 1

[TPD*18] TRAPP M., PASEWALDT S., DÜRSCHMID T., SEMMO A., DÖLLNER J.: Teaching image-processing programming for mobile devices: A software development perspective. In *Proc. Eurographics Education Papers* (2018), The Eurographics Association, pp. 17–24. 3, 7

[VPR13] VONDRICK C., PATTERSON D., RAMANAN D.: Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision 101*, 1 (2013), 184–204. 2, 5

[WP10] WELINDER P., PERONA P.: Online crowdsourcing: Rating annotators and obtaining cost-effective labels. In *Proc. IEEE CVPR Workshops* (2010), pp. 25–32. 2, 6

[ZYC*20] ZOU Y., YANG J., CEYLAN D., ZHANG J., PERAZZI F., HUANG J.: Reducing footskate in human motion reconstruction with ground contact constraints. In *Proc. IEEE WACV* (2020), pp. 448–457. 2