On the Combination of Gamification and Crowd Computation in Industrial Automation and Robotics Applications

Tom Bewley[1] and Minas Liarokapis[2]

***Abstract*— Autonomous intelligent systems outperform human workers in an expanding range of domains, typically those in which success is a function of speed, precision and repeatability. However, many cognitive tasks remain beyond the reach of automation. In this work, we propose the use of video games to crowdsource the cognitive versatility and creativity of human players to solve complex problems in industrial automation and robotics applications. To do so, we introduce a theoretical framework in which robotics problems are embedded into video game environments and gameplay from crowds of players is aggregated to inform robot actions. Such a framework could enable a future of synergistic human-machine collaboration for industrial automation, in which members of the public not only freely offer the fruits of their intelligent reasoning for productive use, but have fun whilst doing so. There is also potential for significant negative consequences surrounding safety, accountability and ethics if great care is not taken in the implementation. Further work is needed to explore these wider implications, as well as to develop the technical theory behind the framework and build prototype applications.**

I. Introduction

Autonomous intelligent systems outperform human workers in an expanding range of domains, typically those in which success is a function of speed, precision and repeatability, but many tasks remain beyond their reach. As humans, we excel at solving problems requiring planning, spatial reasoning, semantic interpretation, social collaboration and creativity. Each of these skills is in high demand in the robotics industry, and each is tested rigorously during video gameplay. By way of example: in the Shakespeare Anthology puzzle of the 2003 survival game Silent Hill 3, the goal is to enter a four-digit code into a keypad. To obtain the numbers, the player must discover a poem scrawled on a nearby scrap of paper, peruse the stanzas to identify cryptic allusions to a variety of Shakespearean plays, and follow a convoluted series of deductions requiring an intimate knowledge of the bards literary output. Such a feat is far beyond the capacities of today's artificial intelligent systems.

One could view the several hundred billion hours spent gaming annually by the world's 2.3 billion players [1][2] as a vast, untapped resource of valuable intelligence. This time is worth upwards of $2 trillion if paid with the median US salary [3], but the gamer community actually pays to participate in the round-the-clock exhibition of complex, subtle reasoning. The legions are connected by robust, high-speed networks, and communicate their intentions precisely via the simple mechanisms of gaming controllers, bypassing the need for expensive, error-prone sensors. Individually, they are incentivised by competition to adapt and learn from mistakes. Collectively, they provide robust, parallel computation. Yet while developers collect gameplay statistics for their own purposes, this perfect problem-solving storm is otherwise entirely wasted from an economic perspective.

Could the robotics industry harness the power of gamer intelligence to address its challenges efficiently and securely? More specifically, is it possible to map the salient aspects of a robotics problem to features of a video game, analyse the relevant in-game actions of one or more players, and reinterpret the most effective strategies into control instructions for the robot itself? This would represent a novel combination of *gamication*, in which game mechanics are used for productive ends, and *crowdsourcing*, the accumulation of knowledge, skills and ideas from crowds of individuals towards a single goal. Crucially, gamification s hould not detract from the fun of gameplay. In fact, players need not be aware that they are solving a robotics problem at all.

Numerous experimental results have attested to the efficacy of crowdsourcing for solving complex reasoning problems, with the approach outperforming experts in domains requiring significant adaptability and semantic understanding [4]. The validity of crowdsourcing is also underpinned by theoretical findings t hat c ollaborative c ontrol b y a diverse ensemble of imperfect agents can be more fault-tolerant than using any single agent [5]. However, we recognise a number of weaknesses in past approaches to crowdsourcing for robotics applications, relating primarily to cost-effectiveness, privacy, scalability and ethics, which may be hampering widespread application in industrial automation scenarios.

The primary aim of this work is to provide a high-level description of how gamification and crowdsourcing could be combined in a synergistic fashion to solve industrial automation and robotics problems. To facilitate further comparisons and development of the proposed framework, we have created a companion repository which introduces a standardised terminology for describing crowdsourcing techniques for robotics and related applications, through analysis and cross-comparison of prior work. The repository also discusses the benefits o f t he p roposed f ramework, e xample applications, the challenges of practical implementation, and the broader commercial and societal implications. It can be found at:

www.newdexterity.org/gamification

[1]Tom Bewley is with the Department of Computer Science, University of Bristol, United Kingdom. Email: tom.bewley.2014@bristol.ac.uk

[2]Minas Liarokapis is with the New Dexterity research group, Department of Mechanical Engineering, University of Auckland, New Zealand. Email: minas.liarokapis@auckland.ac.nz

II. Background and Related Work

Human computation is the technique of outsourcing certain cognitive tasks from machines to humans, to make time-efficient and cost-efficient use of the relative abilities of each. The computation is generally said to be *crowdsourced* if multiple humans participate. Human computation has its origins in work on interactive evolutionary computation, pioneered by Richard Dawkins, in which user preferences were used to guide artificial evolution algorithms in place of fitness functions [6]. Since the advent of the internet, which allows large groups to communicate at virtually zero cost, applications of crowd computation have proliferated under names such as *crowdvoting* to help public-facing organisations make popular branding and strategy decisions [7], *citizen science* in which members of the public assist in data collection and analysis for scientific research [8], and *open innovation* where online groups work together on solutions to large global problems [9]. The premise in each of these cases is that a wide diversity of thought is beneficial, and a superior result can be obtained through the aggregation of many ideas and opinions (in a synergistic fashion) than from any single person acting alone - this concept is often referred to as the *wisdom of the crowd*.

From the perspectives of both simplicity and commercial applicability, some of the most elegant crowdsourcing mechanisms have been the *games with a purpose* of Luis von Ahn [10]. These include *the ESP game* [11], where players are randomly paired and shown a series of images, which they must each independently label with a single word. The pair receives a positive score if their labels match, hence are incentivised to pick words that truly represent the image. The game produces high-quality labels that can be used to improve online image search. In another word-guessing game, *Verbosity* [12], player pairs alternate between the roles of describer and guesser. The describer is provided with a secret word x, and must help the guesser to find it by filling in a series of sentence templates such as "x is a kind of...". The describer can then rate each of the guesser's inputs as "hot" or "cold", and the pair is rewarded based on guessing speed. Over time, through the agreement of many player pairs, the game produces a database of commonsense facts about the secret words (e.g., dog is kind of pet).

Outside of games, von Ahn also developed a tool called *reCAPTCHA* [13], which is used billions of times annually to secure websites from bots by forcing visitors to read and re-type obscured and distorted text. Unbeknownst to most users, this text derives from scans of archived books, and responses are autonomously aggregated to digitise the books for online publication. In 2011 von Ahn founded *Duolingo* [14], a free language-learning platform from which students efforts are similarly harnessed to translate online content.

Numerous prior attempts have been made to apply crowdsourcing to robotic control. In the work of Goldberg et al. in the early 2000s, crowds were provided with live video feeds mounted on or near robotic platforms, and used keyboard and mouse commands to dictate how they wished the robot to move. Specific applications included moving a robot arm over a Ouija board [15], controlling the pan, tilt and zoom of a camera trained on a visual scene of interest [16], and dictating the route of a mobile robot around a museum [17]. Given that user preferences naturally differed, a major technical challenge was the aggregation of many inputs into a single instruction for the robot that best represented the crowd as a whole; approaches included finding the arithmetic mean of continuous inputs, and treating responses as *votes* which could be grouped into clusters and counted to find a majority. Other groups have employed crowdsourcing to segment 3D scenes into semantically-labelled objects, enabling more dexterous robot grasping [18] and to teach a wheeled robot to navigate a maze by repeated demonstration [19].

It must be noted that none of these cases used gamification to incentivise crowd participation. In the robotics field, the closest example has been the development of a two-player online game called *Mars Escape* [20], from which a database of player actions and instant messaging dialogues was collected. Data from hundreds of player pairs were used to train behavioural and language synthesis models for a real-world robot, with the aim of enabling natural human-robot collaboration [21]. This transfer to the real world was identified as highly challenging, since the in-game world needed to accurately mimic the physics and layout of the real environment for the dataset to be applicable.

The term *continuous crowdsourcing* was coined by Walter Lasecki to describe a scenario where a crowd faces a temporal sequence of tasks, as the external environment with which they are interacting changes in real-time. Laseckis *Crowds and Machines* (CROMA) laboratory at the University of Michigan places significant focus on such continuous applications. In recent years, the lab has harnessed crowdsourcing to remotely control software interfaces [22], caption streaming audio for deaf people [23], answer natural language queries about events in live video feeds [24], convert hand sketches into functional user interface designs as they are still being drawn [25], and perform on-the-fly annotation of 3D scenes to enable more informed robot action planning [26]. In continuous crowdsourcing, planning becomes important, posing a challenge when individuals have divergent ideas about what should be done next. In response, CROMA has developed a novel approach to action aggregation, in which a single *leader* is periodically elected based on their historic agreement with the crowd average and only the leader's actions are taken into account. This creates a greater degree of continuity in control, allowing the leader to plan their actions several steps in advance [22]. Leader-based control was found to be the most robust of the options trialled by a different group, which explored the use of crowdsourcing to help visually impaired people navigate public spaces [27].

The majority of prior crowdsourcing efforts have not harnessed gamification, instead opting to pay human participants for their time and recruit via websites such as Amazon Mechanical Turk [28]. Various attempts have been made to augment these platforms with intelligent task allocation and scheduling systems to improve both speed and cost

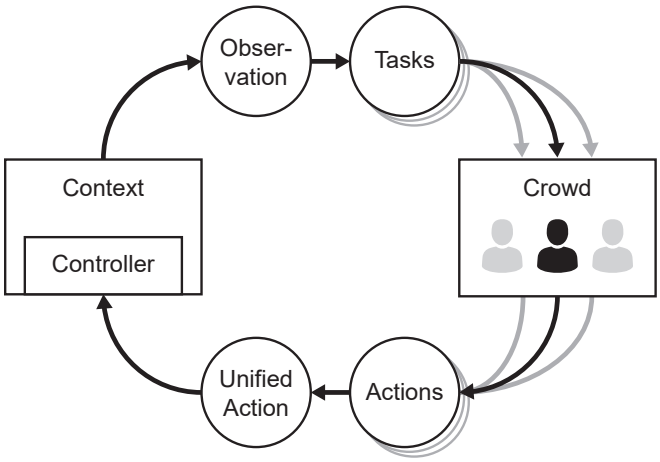

Fig. 1. Cyclic flow of information within a generic crowd computer. A crowd may consist of any number of human players, including just one in the simplest case. If there is only a single player, the unified action becomes identical to that of the player and no aggregation is required. The controller, situated inside the context, may be another human or a purely artificial agent. Player icon sourced from [32].

efficiency. For example, CROMA has developed *Plexiglass*, an algorithm for interleaving multiple tasks into a single user interface [29] and *the lookahead approach* to continuous crowdsourcing [30], which predicts a range of states that may be encountered several seconds into the future, and preemptively asks the crowd to respond, enabling the system to act in real-time when the states actually occur.

Concerns have been raised over the privacy and security risks of crowdsourcing via semi-public platforms such as the Mechanical Turk. Several of the papers cited in this section have paid some attention to this issue, and suggested task information such as images be pre-processed by adding random noise [15], cropping [22], or deleting colour channels [26]. A more principled approach called *Crowdmask* [31] uses crowdsourcing itself to solve the privacy problem. Members of a crowd are exposed to small segments of images and are asked to label areas that may contain parts of private content such as credit card numbers or human faces. By dividing full images between many people, no one individual ever sees truly sensitive material.

The majority of prior work on crowd computer design has assumed that players participate through purpose-built crowdsourcing platforms such as the Mechanical Turk and receive a monetary reward for doing so. We believe that there are numerous advantages to instead embedding human computation tasks in video games, and incentivising players through the enjoyment of gameplay itself. A detailed list of the benefits of gamification is contained in the companion respository that we have developed (benefits section), due to page limit constraints.

III. A Generic Crowd Computer

Throughout the remainder of this paper we use the term *crowd computer* to mean the totality of a system that harnesses crowdsourced human computation to solve a problem, encompassing all hardware and software subsystems, and all human and artificial agents. Despite significant commonalities in the overall structure and workflow of past crowd computer implementations, cross-comparison is made difficult by the historic use of diverse application-specific terminology, as noted in a prior review of the crowdsourcing literature that has been conducted in [33]. In this section, we present a generic, application-neutral crowd computer description, using standard terms (shown in **bold**), which we intend to be malleable enough to describe all prior work in the field. The description of the cycle flow of information of a crowd computer can be best understood with a reference to the generic diagram presented in Fig. 1.

A crowd computer is built to solve a problem in a certain **context**: some physical, social or digital system. Success in solving the problem can be described in terms of a **performance measure** to be optimised. While it is not possible to access the complete state of the context, a partial **observation** can be made. Whenever human computation is required, a context state observation is stored in a data structure, and pre-processed by a function called a **task mapping**. This mapping outputs a set of parameters called a **task**, which in turn is used to modify features of a software environment to which one or more human participants have access. Individually, these participants are called **players**[1], and the complete set of players which sees a particular task is a **crowd**. Depending on the problem to be solved, a unique task mapping function may be used for each player, resulting in task **differentiation** across the crowd.

Tasks within crowd computer persist for a prescribed **interval** of time, during which the role of each player is to produce an appropriate **action** in response. Players synthesise actions by interacting with a set of hardware and software **tools**. During an interval, players may be able to **communicate**, thereby influencing each other's actions. At the end of the interval, player actions are recorded in a prescribed **action format**, and passed through a **validation** step to correct or remove obvious errors. Where the crowd consists of multiple people, validated actions are **aggregated** into a **unified action** through an averaging, voting, summation or leader-election operation. The unified action can be viewed as the crowd computer's output, where its input is the context state observation. The unified action is given to a human or artificial agent situated in the context, called the **controller**, who is influenced by its value but may act with a degree of **autonomy** when choosing how to behave in the context. When the controller makes a decision, it modifies the state of the context, and thus also the performance measure. The performance measure may be fed back to the players, allowing them to observe whether their collective actions were beneficial. State information may also be immediately obtained again, and a new task synthesised, causing the crowd computer to operate iteratively and with **continuity**.

[1]Our adoption of this term may appear to inherently favour the gamification approach to crowdsourcing ("worker" is more common in the crowdsourcing literature), but we believe it draws healthy attention to the importance of the incentives, strategies, biases and subjective experiences of human participants.

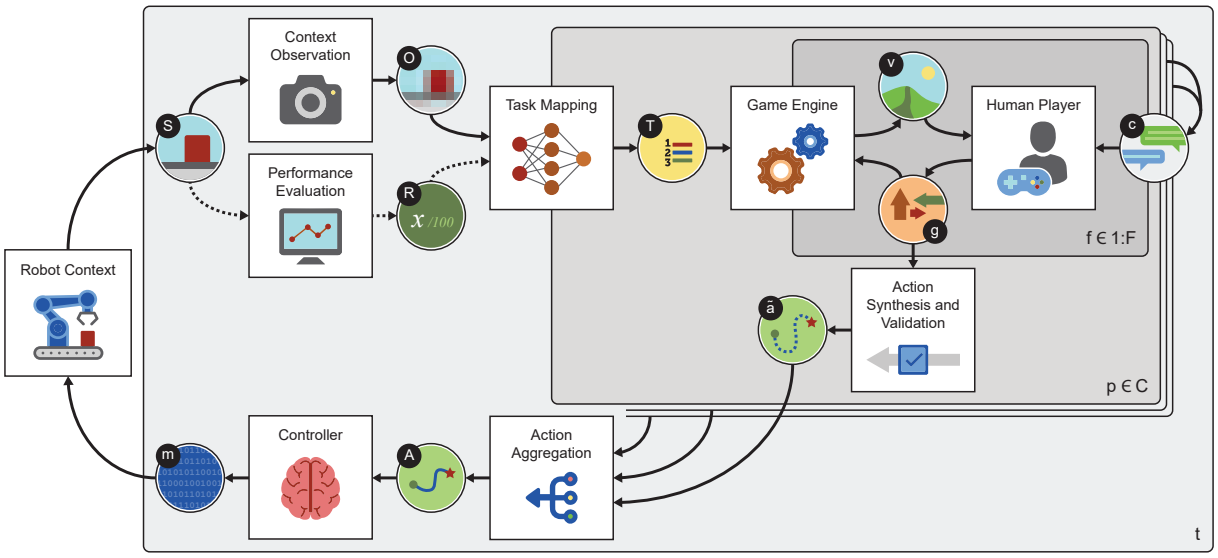

Fig. 2. Flow of information within a gamified crowd computer for industrial automation. Grey plates indicate a nested iteration over the variables indicated in the bottom-right corners. Icons sourced and modified from [32].

IV. Combining Gamification and Crowdsourcing

In this section, we introduce the proposed theoretical framework that combines gamification and crowd computation for industrial automation and robotics applications.

A. Problem Formulation

The following formulation augments the generic crowd computer description from the previous section with several game-specific concepts (*italic*), and introduces more compact mathematical notation. We accept that it is one of many potentially valid formulations; our intention is not to constrain future work to a single narrow approach, but to provide a basis for further discussion.

Robotics problems are inherently dynamic, meaning time is a crucial variable. At a given time t, a robot's **context**, consisting of its physical structure and that of its environment, has a particular state S_t. S_t is not directly visible, but an **observation** $O_t(S_t)$ can be obtained to summarise it, partly via the robot's sensor equipment but potentially also from external sources such as human operatives and technical documents. O_t is passed into a **task mapping** function, producing a set of parameters called a **task** $T_t(O_t)$, which is in turn used to define features of a video game world. The video game is attended by one or more human **players**, collectively forming a **crowd** C, and for each player $p \in C$ the task mapping may be unique: $T_t^{(p)}(O_t)$. This creates **task differentiation**.

Within a single task **interval** $\{t \to t+1\}$, the video game evolves over F states called *frames*, where $F \geq 1$. At each frame f, the game engine first creates a game *visualisation* (which may include any mixture of sensory stimuli) for each p which, in the most general case, is a function of the entire crowd's tasks, $v_{t,f}^{(p)}(T_t^{(C)})$. Players may then **communicate** with each other, which can be modeled as p receiving information $c_{t,f}^{(p)}$ from the rest of the crowd. In tandem with the current visualisation, communication modifies p's *gaming policy* $\pi^{(p)}$, a set of latent, idiosyncratic properties which includes their skills, memories and ongoing plans. p then uses the available **tools** (e.g. gamepad, keyboard, mouse, touchscreen) to generate in-game behaviour known as *gameplay*, which can be viewed as the output of their gaming policy given the current visualisation: $g_{t,f}^{(p)} = \pi^{(p)}(v_{t,f}^{(p)})$. The game visualisation for the next frame is a function of the whole crowd's gameplay decisions.

At the end of a task interval, the encompassed history of p's gameplay is condensed into a prescribed **action format** $a_t^{(p)}(g_{t,1:F}^{(p)})$, which may be passed through a **validation** step to correct errors and remove invalid actions. Validated actions $\tilde{a}_t^{(p)}$ from the entire crowd are **aggregated** into a **unified action** $A_t(\tilde{a}_t^{(C)})$. A_t is sent to the robot's internal software — the **controller** — which takes the value as a high-level instruction but may act with a degree of **autonomy** in choosing a particular sequence of motor commands $m_t(A_t)$. Executing m_t produces physical motion, thereby modifying the state of the context for the next task interval $S_{t+1}(m_t)$ and leading to **continuity** over time.

For industrial automation applications, a quantitative **context performance measure** can generally be sought. This measure R may not be evaluated at every task interval, but when it is, it can be fed back into the task mapping function, meaning that in general, tasks are functions of both the context observation and the latest performance measure: $T_t^{(p)}(O_t, R_t)$. Giving players access to R, which may be practically represented as an on-screen score, provides **context feedback** and allows players to adjust their gaming policies in response. The above formulation is represented visually in Fig. 2 (an extended and less abstract version of Fig. 1) which summarises the core structure of the proposed theoretical framework. Intra-crowd communication and measurement of context performance are added, along-

side domain-specific features such as a game engine which generates a visualisation for each player at each frame. It should be stressed, however, that the high-level cyclic information flow from context to player and back via the intermediate variables of observation, task, action and unified action remains unchanged. This information flow is at the core of any crowd computer.

The formulation is also summarised as pseudo-code in Algorithm 1 that is presented below. Operations that must be completed by bespoke crowd computation software are represented by black imperative statements. Operations completed by human gamers or within the robot's context are represented by grey declarative statements. The designer of a crowd computer would have no direct control over the latter kind of operation.

Algorithm 1: Gamification-Based Crowd Computation for Use in Both Industrial Automation and Robotics Applications

Robot context adopts initial state S_0;
foreach t **do**
  Observe context state $O_t(S_t)$;
  if *context performance being evaluated* **then**
    Evaluate context performance $R_t(S_t)$;
  else
    $R_t = []$;
  end
  Assemble crowd C;
  for $p \in C$ **do**
    Map into task parameters $T_t^{(p)}(O_t, R_t)$;
    Initialise gameplay history $g_{t,0}^{(p)} = []$;
    for $f \leftarrow 1$ **to** F **do**
      if p *active on game* **then**
        Create game visualisation $v_{t,f}^{(p)}(T_t^{(C)}, g_{t,f-1}^{(C)})$;
        Player communicates $c_{t,f}^{(p)}$;
        Player updates gaming policy $\Delta\pi^{(p)}(v_{t,f}^{(p)}, c_{t,f}^{(p)})$;
        Player engages in gameplay $g_{t,f}^{(p)} = \pi^{(p)}(v_{t,f}^{(p)})$;
      else
        $g_{t,f}^{(p)} = []$;
      end
    end
    Condense gameplay into action $a_t^{(p)}(g_{t,1:F}^{(p)})$;
    Validate and attempt to correct action $\tilde{a}_t^{(p)}(a_t^{(p)})$;
    if $a_t^{(p)}$ *cannot be corrected* **then** delete it;
  end
  Aggregate into unified action $A_t(\tilde{a}_t^{(C)})$;
  Controller produces motor commands $m_t(A_t)$;
  Context state evolves $S_{t+1}(m_t)$;
end

B. Task Mapping

In the majority of prior work on crowd computation, context state observations O have either been presented directly to players or subject to only minor pre-processing (e.g. cropping and filtering in the case of images, as presented in [24]). In our formulation, however, we refer to a task mapping function $T^{(p)}(O, R)$ which applies a transformation to O and, when available, the context performance measure R, before embedding it in a game environment. Consideration of this step will be crucial for the successful implementation of crowd computers from the perspectives of both privacy and player enjoyment. Consider a manufacturer that wishes to build a crowd computer to assist a robot in navigating a busy factory floor. With the context state S comprising the robot and its immediate environment, O could be provided by an onboard video camera and a worded instruction such as *"go to the box containing component A while avoiding obstacles"*. A typical crowd computer would show these data directly to players, but what if the factory, and component A in particular, contained valuable trade secrets crucial to the manufacturer's success? Additionally, what if the task of locating A were so cripplingly dull that nobody would wish to play a video game involving it? In the former case, the operation of the crowd computer would result in severe breaches of privacy and security, and in the latter, there would soon be no players to make it operate at all. Task mapping serves to rectify these issues by transforming O and R into a task T that is less clearly connected with the context and is more enjoyable for players. For example, the private and uninspiring visual scene of the factory floor may be transformed into a haunted forest in a much larger fantasy game world. In the transformed task, the required sequence of player actions a may be identical, but the manner in which they are incited is far more engaging. In the process of task mapping, significant information may be added to decorate the game world with interesting features, as long as these do not interfere with the task completion. Conversely, a large portion of the information present in O may be stripped away, leaving only a simplified, coarse-grained representation of the underlying robotics problem. For some applications this may be enough for the crowd computer to remain effective.

One pertinent question is: how obscure could a task mapping be without making the resultant game environment entirely irrelevant to the robotic context? Would spatial features such as the locations of objects or a robot arm's joint positions need to be mapped into similarly-arranged spatial features in the game world, or could they instead parametrise completely different structures or even the behaviours of in-game characters? Potentially if an equally complex *inverse* mapping from gameplay g to actions a was employed, and the two mappings were optimised in parallel, it could be possible to recover contextually-relevant robot control instructions regardless how indirect the isomorphism between the context and the game is. Further work is needed to investigate this prospect.

C. Context Physics Modelling

If for each task interval within a game-based crowd computer, $F > 1$, then the video game visualisation must be rendered over numerous frames between observations of the context state. This fundamentally modifies the problem from that of one-task, one-action (as in most prior work) to a situation where the crowd's response to each task is a *temporal sequence* of gameplay. Crucially, in order for this sequence to be applicable in the context, the internal physics of the game must mimic the physics of the context and the cause-effect relationships must track each other over time. If, on the other hand, $F = 1$, the context state is measured before every frame, so it can be fed immediately into the game visualisation. In the latter case, the game engine can be entirely ignorant of the physics of the context, representing whatever information it receives, one frame at a time.

To illustrate this point more concretely, consider the task of controlling a robot arm to pick up a complex object. If only one observation of the context state (such as the robot's joint angles and the shape and relative position of the object in question) were recorded at the initialisation of the task, the video game would require a *model* to simulate how the robot and object would be moved by a sequence of player commands over a period of several seconds. Since each player would likely input a slightly different set of commands, the game simulation would proceed differently for each. Alternatively, if the robots were being controlled by a crowd in real-time, and the same state information was observed every 40 milliseconds, a game running at 25 frames per second could simply use the latest joint angles and object positions to render visualisations in a completely model-free manner. Importantly, in a model-free crowd computer, the visualisation for every player in the crowd would show the effects of *actual* robot motion as guided by the unified action, rather than that which would have resulted from their actions alone, so clever game mechanics would need to be developed to ensure that a confusing dissonance between behaviour and outcome is not experienced.

D. Feedback, Scoring and Learning

A number of prior works [22][24][26][34] have discussed the possibility of using a crowd computer to generate a large dataset for training a supervised machine learning model, to which tasks may be gradually *handed off* once its performance is adequate, enabling faster response times than through continual reliance on human computation. We endorse this bootstrapped approach to human-machine collaboration, but we believe that the incorporation of machine learning into crowd computers could be far deeper, namely to improve the system itself and optimise the utility of output actions with respect to the context performance measure.

In a video game, players generally act to maximise their score. While it may be desirable to simply use the context performance measure as the game score, thereby directly tying in-game performance to robot performance, it is unrealistic to expect this measure to be accessible at every task interval. An artificial score must therefore be synthesised to incentivise gameplay that produces good performance in the context. Successfully hand-coding a system where this condition is met, may be an extremely challenging exercise, especially where the task mapping is complex. Instead, we envisage a dual usage of the context performance measure: firstly to provide direct feedback to players when available, and secondly to inform machine learning to optimise internal parameters of the crowd computer – such as the task mapping function – so that in-game performance and robot performance become increasingly aligned over time.

This particular problem may be a good candidate for *reinforcement learning* (RL) approaches. At the outermost level, the objective of a crowd computer is to take in noisy and incomplete context state observations O and output appropriate actions A, while occasionally receiving performance feedback R. This bears more than a passing resemblance to the typical formulation of a *partially-observable Markov decision process* (POMDP), which RL is commonly employed to solve [35]. Just as the crowd has a set of gaming policies $\pi^{(C)}(g|v)$ one might imagine these being subsumed by a *whole-system policy* $\Pi(A|O)$, which adds elements such as task mapping, action aggregation and the in-game physics model (if $F > 1$), and which could be amenable to gradual refinement given enough data. This opens up an intriguing prospect: now that the viability of RL for video game *play* has been successfully demonstrated [36][37], could it be possible to invert the problem and use RL for game *design* with the goal of inducing useful patterns in human play? Clearly, such unconventional use of RL would only be justified if learning Π was more tractable than learning to control a robot directly. There are reasonable grounds to expect this to be the case, as learning a problem representation that is comprehensible by highly-versatile humans intuitively seems like half the work compared with solving the problem in full.

V. Conclusion

We have presented a theoretical framework for combining gamification and crowd computation to assist robotic systems in industrial contexts and complex robotics applications, discussing its potential benefits relatively to the more common crowdsourcing approaches and outlining a possible technical framing of the problem. On the companion repository, we have presented two illustrative examples of how the framework could be implemented in practice, and we have discussed possible benefits of the approach as well as some of the broader commercial and societal implications.

Gamification can improve the cost-effectiveness, privacy, scalability and ethical integrity of the crowdsourcing industry, and has numerous features that make it especially suitable for robotics applications. If a platform can be developed to deliver game-based crowd computation in a manner that is secure, efficient and fair to all stakeholders, we envisage a future of flexible, synergistic human-machine collaboration for industrial automation, in which members of the public not only freely offer the fruits of their intelligent reasoning for productive use, but have fun whilst doing so.

References

[1] Limelight Networks, "The state of online gaming," https://www.limelight.com/blog/state-of-online-gaming-2018/, 2018, accessed: 28 Feb 2019.

[2] Newzoo, "2018 global games market report," 2018.

[3] Bureau of Labor Statistics, United States Department of Labor, "Usual weekly earnings of wage and salary workers fourth quarter 2018," https://www.bls.gov/news.release/archives/wkyeng_01172019.htm, 2018, accessed: 28 Feb 2019.

[4] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng, "Cheap and Fast But is it Good? Evaluating Non-Expert Annotations for Natural Language Tasks," *Conference on Empirical Methods in Natural Language Processing (EMNLP '08). Association for Computational Linguistics, Stroudsburg, PA, USA, 254-263*, 2008.

[5] K. Goldberg and B. Chen, "Collaborative control of robot motion: robustness to error," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 655–660, 2001.

[6] R. Dawkins, *The Blind Watchmaker*. Norton & Company, 1986.

[7] V. F. Araman and R. Caldentey, "Crowdvoting the timing of new product introduction," Jan 2016. [Online]. Available: http://dx.doi.org/10.2139/ssrn.2723515

[8] R. Bonney, C. B. Cooper, J. Dickinson, S. Kelling, T. Phillips, K. V. Rosenberg, and J. Shirk, "Citizen science: A developing tool for expanding science knowledge and scientific literacy," *BioScience*, vol. 59, no. 11, pp. 977–984, 2009.

[9] K. R. Lakhani, A.-L. Fayard, N. Levina, and S. H. Pokrywa, "OpenIDEO," *Harvard Business School Technology & Operations Mgt. Unit*, no. Case No. 612-066, Feb 2012. [Online]. Available: https://ssrn.com/abstract=2053435

[10] L. von Ahn and L. Dabbish, "Designing games with a purpose," *Communications of the ACM*, vol. 51, no. 8, Aug 2008. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1378704.1378719

[11] L. von Ahn and L. Dabbish, "Labeling images with a computer game," *Proceedings of the 2004 conference on Human factors in computing systems - CHI 04*, pp. 319–326, 2004.

[12] L. von Ahn, M. Kedia, and M. Blum, "Verbosity: a game for collecting common-sense facts," *Proceedings of the SIGCHI conference on Human Factors in computing systems - CHI 06*, p. 75, 2006.

[13] L. von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum, "reCAPTCHA: Human-Based Character Recognition via Web Security Measures," *Science*, vol. 321, no. 5895, pp. 1465–1468, 2008. [Online]. Available: http://science.sciencemag.org/content/321/5895/1465

[14] "Duolingo," https://www.duolingo.com/, accessed: 23 Aug 2018.

[15] K. Goldberg, B. Chen, R. Solomon, S. Bui, B. Farzin, J. Heitler, D. Poon, and G. Smith, "Collaborative teleoperation via the internet," *IEEE International Conference on Robotics and Automation*, pp. 2019–2024, 2000.

[16] D. Song, A. Pashkevich, and K. Goldberg, "ShareCam part II: approximate and distributed algorithms for a collaboratively controlled robotic Webcam," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1087–1093, 2003.

[17] K. Goldberg, D. Song, Y. Khor, D. Pescovitz, A. Levandowski, J. Himmelstein, J. Shih, A. Ho, E. Paulos, and J. Donath, "Collaborative online teleoperation with spatial dynamic voting and a human tele-actor," *IEEE International Conference On Robotics And Automation*, pp. 1179–1184, 2002.

[18] A. Sorokin, D. Berenson, S. S. Srinivasa, and M. Hebert, "People helping robots helping people: Crowdsourcing for grasping novel objects," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2117–2122, 2010.

[19] S. Osentoski, G. Jay, C. Crick, and O. C. Jenkins, "Crowdsourcing for closed-loop control," *Proc. of the NIPS Workshop on Computational Social Science and the Wisdom of Crowds, NIPS*, 2010.

[20] S. Chernova, J. Orkin, and C. Breazeal, "Crowdsourcing HRI Through Online Multiplayer Games," *AAAI Fall Symposium Series*, 2010.

[21] S. Chernova, N. DePalma, and C. Breazeal, "Crowd-sourcing real-world human-robot dialogue and teamwork through online multiplayer games," *AI Magazine*, vol. 32, no. 4, pp. 100–111, Dec 2011.

[22] W. S. Lasecki, K. I. Murray, S. White, R. C. Miller, and J. P. Bigham, "Real-time crowd control of existing interfaces," *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pp. 23–32, 2011.

[23] W. Lasecki, C. Miller, A. Sadilek, A. Abumoussa, D. Borrello, R. Kushalnagar, and J. Bigham, "Real-time captioning by groups of non-experts," *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pp. 23–24, 2012.

[24] G. Laput, W. S. Lasecki, J. Wiese, R. Xiao, J. P. Bigham, and C. Harrison, "Zensors: Adaptive, rapidly deployable, human-intelligent sensor feeds," *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 1935–1944, 2015.

[25] W. S. Lasecki, J. Kim, N. Rafter, O. Sen, J. P. Bigham, and M. S. Bernstein, "Apparition: Crowdsourced user interfaces that come to life as you sketch them," *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 1925–1934, 2015.

[26] S. R. Gouravajhala, J. Yim, K. Desingh, Y. Huang, O. C. Jenkins, and W. S. Lasecki, "Eureca: Enhanced understanding of real environments via crowd assistance," *Sixth AAAI Conference on Human Computation and Crowdsourcing*, 2018.

[27] W. L. Khoo, G. Olmschenk, Z. Zhu, and T. Ro, "Evaluating crowd sourced navigation for the visually impaired in a virtual environment," *IEEE International Conference on Mobile Services*, pp. 431–437, 2015.

[28] "Amazon Mechanical Turk," https://www.mturk.com/, accessed: 23 Aug 2018.

[29] A. Rao, H. Kaur, and W. S. Lasecki, "Plexiglass: Multiplexing passive and active tasks for more efficient crowdsourcing," *In Sixth AAAI Conference on Human Computation and Crowdsourcing*, 2018.

[30] A. Lundgard, Y. Yang, M. L. Foster, and W. S. Lasecki, "Bolt: Instantaneous crowdsourcing via just-in-time training," *Proceedings of the CHI Conference on Human Factors in Computing Systems*, no. 2, 2018.

[31] H. Kaur, M. Gordon, Y. Yang, J. P. Bigham, J. Teevan, E. Kamar, and W. S. Lasecki, "Crowdmask: Using crowds to preserve privacy in crowd-powered systems via progressive filtering," *Fifth AAAI Conference on Human Computation and Crowdsourcing*, 2017.

[32] "Various icons from Flaticon.com. Robot arm, computer monitor, brain, gears, user and checkbox icons designed by Freepik. Camera icon designed by Daniel Bruce. Merging arrows icon designed by Smashicons. Gamepad icon designed by fjstudio." https://www.flaticon.com/, accessed: 28 Feb 2019.

[33] M. Hosseini, A. Shahri, K. Phalp, J. Taylor, and R. Ali, "Crowdsourcing: A taxonomy and systematic mapping study," *Computer Science Review*, vol. 17, pp. 43–69, Aug 2015.

[34] G. R. Calegari, G. Nasi, and I. Celino, "Human computation vs. machine learning: an experimental comparison for image classification," *Human Computation*, vol. 5, no. 1, pp. 13–30, 2018.

[35] M. Spaan, *Partially Observable Markov Decision Processes*. in Reinforcement Learning. Adaptation, Learning, and Optimization, vol. 12. Springer, 2012.

[36] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. P. Lillicrap, K. Simonyan, and D. Hassabis, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *CoRR*, vol. abs/1712.01815, 2017. [Online]. Available: http://arxiv.org/abs/1712.01815

[37] "OpenAI Five," https://blog.openai.com/openai-five/, 2018, accessed: 28 Aug 2018.