

Visual Detection of Unknown Objects in Video Games Using Qualitative Stability Analysis

Xiaoyu Ge, Jochen Renz, and Peng Zhang

Abstract—Many current computer vision approaches for object detection can only detect objects that have been learned in advance. In this paper, we present a method that uses qualitative stability analysis to infer the existence of unknown objects in certain areas of the images based on gravity and stability of already detected objects. Our method recursively searches these areas for unknown objects until all detected objects form a stable structure or no new objects can be identified anymore. We evaluate our method using the popular video game *Angry Birds*. We only start with detecting the green pigs and are able to automatically identify and detect all essential game objects in all 400+ available levels. All objects can be accurately and reliably detected. Our method can be applied to other video games where objects obey gravity and are bound by polygons.

Index Terms—Computer vision, knowledge representation, object detection, qualitative spatial reasoning, stability reasoning.

I. INTRODUCTION

OBJECT detection is an important problem in computer vision which remains unsolved in its generality. With sophisticated methods based on edge detection, color clustering, or key features available, it is possible to achieve reasonable accuracy in detecting previously learned objects [1], [2]. However, detecting unknown objects that have not been seen and learned before remains a challenge [3], [4]. Objects in images, and particularly in real-world images, are often not uniform in color and do not have unique edges that bound the object, but typically have a considerable number of detected edges that are unrelated to the objects' boundary. Therefore, it is very hard to identify what defines an object, which pixels are part of the object, and where the object boundaries are located.

The approach we present in this paper for detecting unknown objects is not based on any methods typically used in computer vision, but uses qualitative spatial relations between already detected objects in order to infer the existence of undetected objects. Our method relies on a qualitative stability analysis and is based on the assumption that objects cannot float in the air without support from other objects. Whenever we detect an object that appears unsupported, we assume that there must be a yet undetected object that supports it. Using our qualitative stability analysis, we know where supporting objects could be located and search these areas for potential new objects

using existing computer vision methods. As such, our method is an extension of existing computer vision methods that can be added to any existing methods and will likely improve detection of yet unknown or unidentified objects. But this also means that we suffer from similar weaknesses as the existing methods and can only detect objects for which we can actually extract visual cues in the image. Consequently our method works best in cases where the unknown objects could be reliably detected if they were known.

The main contribution of this paper lies in providing a novel method for inferring the existence of undetected objects. We initially implement and evaluate our method for 2-D video games, such as the popular games *Angry Birds* or *Candy Crush*, where all foreground activities occur in the same image plane. Video games have the advantage that all images are generated and rendered using computer graphics where objects typically do not have the complexity and diversity that can be found in real-world images. Therefore, it is possible to detect known objects in a reliable way. The initial restriction to 2-D video games serves as a proof of concept and allows us to evaluate our method on its own, independent of other open computer vision problems such as reliably detecting the 3-D shape of objects or dealing with occlusion. While this limits the immediate applicability of our approach to real-world scenes, there is a considerable interest in developing reliable object detection in video games [5], [6].

Angry Birds, for example, is a video game that has obtained increased interest within the AI community in recent years, due to the challenging nature of the problem despite its simple game play. There is an ongoing competition [7] with the goal of developing AI agents that can play new levels better than the best human players. The competition organizers provide a software package including a computer vision module that is able to reliably detect and classify known and previously learned objects, but is unable to detect any unknown objects. In order to be able to solve new levels, which is the declared aim of the competition, an AI agent has to be able to detect new and unknown objects and to learn their physical properties. Using our method, we are able to reliably detect unknown objects in *Angry Birds*. We show that we can detect all objects that the existing computer vision module can detect and with the same accuracy.

In addition, we can detect a large number of unknown objects and automatically classify them into new object categories. We are able to achieve this with minimal prerequisites: We only assume that we are able to detect the green pigs; everything else is unknown. Once all relevant game objects are identifiable, we could then start learning their physical properties, for example, by shooting birds at them and observing how they behave when

Manuscript received January 16, 2015; revised August 22, 2015; accepted November 15, 2015. Date of publication December 08, 2015; date of current version June 14, 2016.

The authors are with the the Research School of Computer Science, Australian National University, Canberra, A.C.T. 0200, Australia (e-mail: fantastdd@gmail.com).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TCIAIG.2015.2506741

hit. As such our method is an important step toward building AI agents that are able to autonomously solve any *Angry Birds* level, or other physics-based video games.

The paper is structured as follows. We first introduce the relevant background on computer vision and on qualitative stability analysis. We then present an improved way of computing stability and discuss the effect it has on detecting unknown images. In Section IV, we present our method of detecting unstable objects and how to find unknown objects that support unstable objects. In Section V, we evaluate our method by applying it to a large number of *Angry Birds* levels with many new objects and different backgrounds, as well as to other games. Section VI summarizes our results.

II. BACKGROUND AND RELATED WORK

A. Object Recognition

Object recognition has two general categories [8]: recognizing a particular object (e.g., John's face) and recognizing generic categories (e.g., cars). There are mainly two major approaches to deal with object recognition, namely appearance-based methods and feature-based methods [9]. Appearance-based methods such as [10] basically record known objects as templates and perform recognition based on the template data set. Feature-based methods recognize objects based on different categories of features such as boundary or size of an object or its colors [11]. Both methods require prior knowledge to discover new objects in a scenario, whereas none of them can tell if the detected objects are interesting key objects or unimportant objects from the background. For example, the vision system [12] currently provided for the *Angry Birds* AI Competition framework is developed on a feature-based method. It takes hard-coded colors and shapes of objects as features and can successfully detect and classify known objects. Unsurprisingly, it lacks the ability of discovering new objects whose features are not included in the existing feature set; however, new objects occasionally appear in the game.

Detection proposal methods (for a survey, see [13]) have been widely applied to object detection. The methods generate proposals based on a diverse set of cues to guide the search for objects. A proposal is a region in the image that is likely to contain objects. Guided by the generated proposals, the object detection algorithm can avoid the exhaustive search by first examining the proposed regions, which achieves computational efficiency. However, the methods are not applicable to our problem. First, detection proposals are based on the assumption that foreground objects can be distinguished from background objects by certain common visual properties. However, it is not always the case in our problem setting where background objects can share the same visual properties with foreground objects. Further, the methods cannot be easily generalized to an unknown environment. Most of the methods rely on an intensive training procedure. They are unable to deal with unknown objects of which the visual properties are not captured by the training data. The proposed method could be regarded as a detection proposal method in the sense that it generates proposed regions that are likely to contain unknown

objects based on stability analysis. Another related area is object localization [14]–[16] that aims at accurately locating the detected objects and distinguishing foreground objects from the background clutter.

Discovering unknown objects and object categories (also known as category learning) is very challenging. The problem can be tackled by supervised learning that requires manual annotations [17] or unsupervised techniques [18] which assumes no prior information. The quality of the unsupervised techniques fundamentally depends on how they determine the similarity of image regions. Some techniques are based on analyzing contexts to discover the unseen object categories. For example, Lee and Grauman [4], [19] use object graphs to model the topological relations of the regions in an image and group the regions that have similar topological relations with the surrounding known objects. However, the accuracy will drop significantly when multiple unseen objects appear in an image while there are only a few known objects.

1) *Object Recognition in Video Games*: There is some work on developing general intelligent agents to play Atari 2600 games (Atari-GGP) [20]. Identifying game objects is one important capability for agents to recognize the game environment (some agents [21], [22] based on deep reinforcement learning do not perform objects identification). Hausknecht *et al.* [23] implemented an algorithm based on artificial neural networks and the algorithm uses different state representations. One state representation relies on a set of game objects that are identified by template matching. In [24], the agent tracks the blobs of pixels that have similar colors across video frames and then obtains game objects by merging the blobs according to certain cues. Since the method categorizes the detected objects by their shape, it may group different objects that have similar shapes but different colors into one category.

B. Structural Stability

Stability of a given structure can be exactly calculated if all relevant physical parameters of all the involved objects, such as shape, density, mass, mass distribution, material are known. Blum *et al.* [25] developed a force-based method to calculate the stability of an object by taking detailed quantitative physical parameters as input. The result is very accurate, however most of the required input information is unavailable, when the only information we have about a structure is what a vision system sees. If less information is available, structural stability can only be approximated. A recent qualitative approach [26] approximates structural stability of a 2-D structure based on the center of mass of rectangular shaped objects. It encodes stability rules using qualitative spatial relations from the extended rectangle algebra (ERA). Each rectangular object can be mapped to a pair of intervals by projecting the rectangle to the x - and y -axes. Qualitative relations between two objects can then be expressed as the interval relations between their corresponding intervals. ERA contains 27 interval relations (Fig. 1) in each dimension instead of the typical 13 interval relations in the original rectangle algebra [27], [28]. The additional relations are obtained by also considering the center point of intervals rather than only their two end points. This extension allows us to consider the


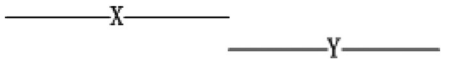
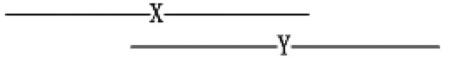
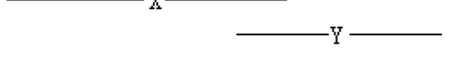
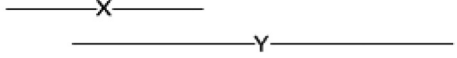

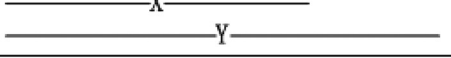
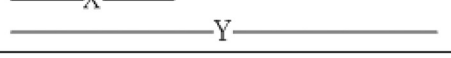
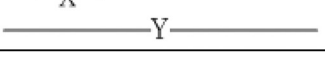
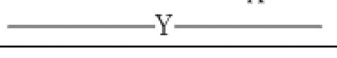
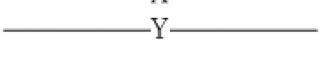
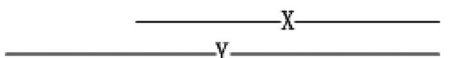
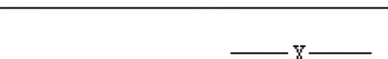
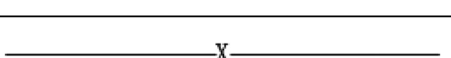
Relation	Illustration	Interpretation
$X b Y$ $Y a X$		X takes place before Y
$X m Y$ $Y mi X$		X meets Y (i stands for <i>inverse</i>)
$X mom Y$ $Y momi X$		most part of X overlaps with most part of Y
$X lol Y$ $Y loli X$		less part of X overlaps with less part of Y
$X mol Y$ $Y moli X$		most part of X overlaps with less part of Y
$X lom Y$ $Y lomi X$		less part of X overlaps with most part of Y
$X ms Y$ $Y msi X$		X starts Y and cover most part of Y
$X ls Y$ $Y lsi X$		X starts Y and cover less part of Y
$X ld Y$ $Y ldi X$		X during left part of Y
$X rd Y$ $Y rdi X$		X during right part of Y
$X cd Y$ $Y cdi X$		X during Y and the midperpendicular of Y through X
$X mf Y$ $Y mfi X$		X finishes Y and cover most part of Y
$X lf Y$ $Y lfi X$		X finishes Y and cover less part of Y
$X eq Y$		X is equal to Y

Fig. 1. The 27 interval relations of ERA.

mass center of rectangles and, therefore, allows for more flexible reasoning under physical constraints [Fig. 2(a) and (b)]. The basic idea to test the stability of an object in [26] is that if the vertical projection of the objects mass center falls into its supporting area, the object is determined as stable, where the *support area* is defined as the horizontal interval between the leftmost and rightmost support points of an object or a structure.

Although ERA is useful to determine the areas for searching new objects when there are few detected objects in the scene, it only roughly approximates the stability of a single object rather than a structure.

In contrast, Gupta *et al.* [29] suggest a method to check the stability of an object by qualitatively analyzing the forces acting on the object. However, this method is also not applicable

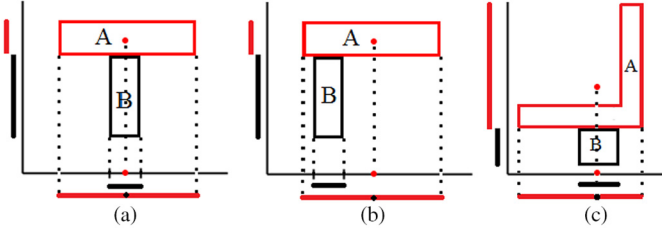


Fig. 2. Local stability determined with ERA. (a) and (c) Locally stable case. (b) Locally unstable case.

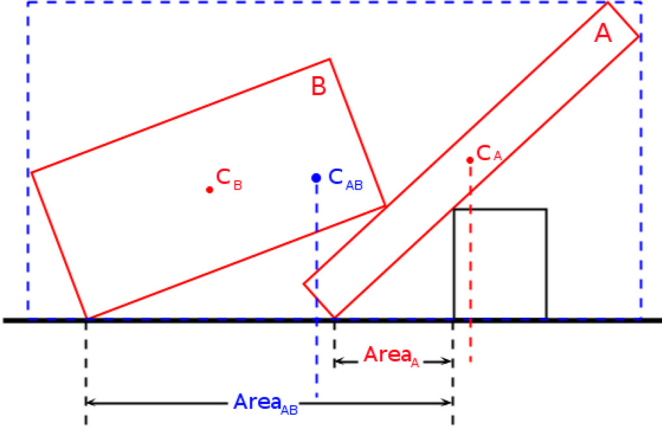


Fig. 3. Configuration where an object is lower (in terms of the top point or the bottom point) than its supporter. C_A and C_B are mass centers of objects A and B, and C_{AB} is the mass center of the combination of A and B. $Area_A$ is the support area of A and $Area_{AB}$ is the support area of the combination of A and B.

for determining structure stability when using only visual input. Some methods have been developed to deal with the stability of the whole structure to achieve a better understanding of a scene. For example, Zheng *et al.* [30] evaluate the relative structural stability by comparing the potential energy of different combinations of objects in the structure. As relative stability is used in this method, it is not helpful for discovering hidden or missing objects.

Differently, Jia *et al.* [31] test the stability of the whole structure by iteratively calculating the mass center of a set of objects from top to bottom and checking if the vertical projection of the mass center falls into the set of objects' supporting area. This method works in simple scenarios (e.g., a stack of several books on a desk); however, as it simply uses a top-down strategy, some supporting relations may not be detected correctly, i.e., not all supporters are lower than their supportees either in case of top point or center point, hence, sometimes it will attempt to evaluate an object before its supportee which may result in an incorrect judgment.

For example, in Fig. 3, object A will be first evaluated using a top-down schema and determined as unstable; however, due to the effect of its supportee B, it can actually remain stable. In Section III-B, we propose an improved method which evaluates objects in an appropriate sequence in terms of their supporting relations.

There are also some methods which use probabilistic simulations to solve physical reasoning problems such as predicting the stability [32]. The method works well when the environment

is fully observable and the physical properties of objects are completely known. However, it is often the case that the environment is partially known and the information about the objects is incomplete. The stability analysis algorithm proposed in this paper is a qualitative approach that can effectively deal with an environment of this nature.

III. QUALITATIVE STABILITY ANALYSIS

The basic assumption we make is that the image we analyze depicts a stable scenario, that is all objects in the image are stable and none of the objects are currently in flight, falling, or otherwise unsupported. We also assume that one or more objects in the image are already known and have been detected. A major limitation we have is that we do not know the physical properties of any of the objects in the image. What is most important in our analysis is that we do not know the density or mass of objects, neither absolute nor relative to each other. This means that even if we knew the exact shape and extent of all objects, it would be impossible to accurately calculate whether a group of objects is stable. Identifying the exact physical properties would require us to actively interact with objects, which is outside the scope of this paper. In the following, we assume that the density of all objects is the same and uniformly distributed. Under this assumption, it is possible to calculate the mass center of an object and approximate the stability of an object in a structure. The important fact to keep in mind is that stability can only be approximated. We will now look at ways to estimate whether a structure is stable.

A. Local Stability

We can use ERA to calculate stability of rectangular objects. However, as ERA only considers rectangular objects, Zhang and Renz [26] assume that the center point of the minimum bounding rectangle (MBR) of an object corresponds to the mass center of the actual object, which in turn corresponds to the center point of the intervals. When objects other than rectangles are used, this correlation between the mass center of an object and center point of its MBR does not hold anymore. However, even when considering objects other than rectangles, for example arbitrary polygons, we can still use ERA to reason about stability of an object. This is because we get the same 27 ERA interval relations if we consider a different point inside the intervals and not their center point [Fig. 2(c)]. Therefore, we can use the actual center of mass of an object and use its corresponding projection to define the ERA relations. Since this method determines stability of an object only with respect to how its center of mass relates to its supporting area, and does not consider what happens above an object, we call this approximation to the stability of an object *Local Stability*. Fig. 2 demonstrates two examples of how ERA works on determining local stability of an object. Specifically, the ERA relation between block A and B in the *x*-axis in the left figure is *center_during* which means A is well supported by B and A is locally stable; in the right figure, the ERA relation between blocks A and B in the *x*-axis is *left_during* which means that the mass center of block A is not supported, thus it is not locally stable.

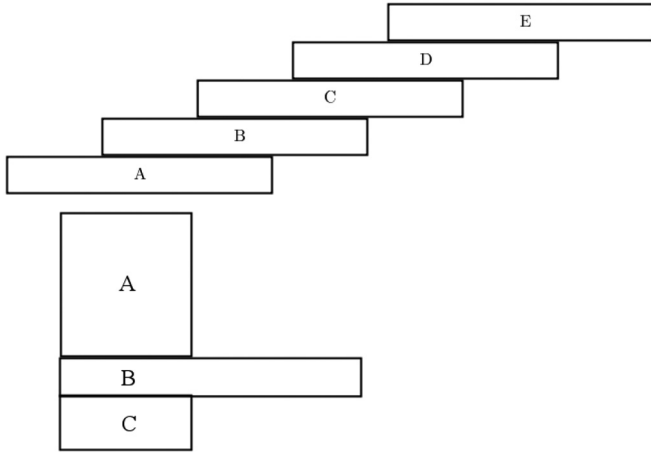


Fig. 4. (a) Configuration that is globally unstable but locally stable: The center of mass of B - E is outside the support area provided by A . (b) Configuration that is globally stable but locally unstable: The center of mass of B is outside the support area provided by C .

Although local stability is only a rough approximation, we can still use it to determine which areas to search for new objects, because it guarantees to search every area that can contain supporters of an object. The specific method to decide searching areas will be introduced in Section IV.

B. Global Stability

It is clear that local stability can give a wrong result depending on the objects above the object we consider. For example, in Fig. 4(b), if we only consider block B and its supporter C , then B is not stable. But if we take the mass of A into account, then B will probably be stable. In this case, local stability provides a false negative result, i.e., it judges an object to be unstable which should be stable. Likewise, we can obtain a false positive result where an object is considered locally stable when it should be unstable [see Fig. 4(a)]. Thus, this method may not be a very good approximation of the real stability of a structure. We propose a stability method that is able to give a better approximation of the stability of a structure compared to local stability by also considering what is above an object. Our method, which we call *global stability* takes as input a labeled directed graph where there is a node for each detected object and a directed edge to specify the supporting relation between two connected objects. We call this the support graph (SG) of a structure (see Fig. 5 for an example). Given an SG, if there exists node N_1 with an edge pointing to node N_2 , then N_1 supports N_2 .

Definition III.1 (Support, Support Depth, Supportees, Direct Supporter): Given an SG, if there is a path from N_i to N_j , then N_i supports N_j . Support depth $SD(N_i, N_j)$ is the length of the shortest path from N_i to N_j . A direct supporter of an object N_j is an object N_i such as $SD(N_i, N_j) = 1$. The supportees of N_i are the set of all nodes that are supported by N_i .

With a support graph, the supportees of an object O can be considered when testing its stability. Specifically, when querying the stability of object O , we will take O and all its supportees as a substructure S and test if the mass center of S falls into the support area that includes all direct supporters of S . Algorithm 6 gives the global stability test.

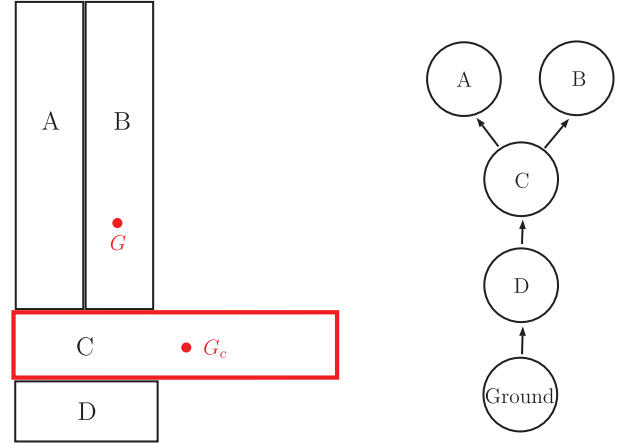


Fig. 5. Globally stable structure and its support graph. G_c indicates the mass center of block C , G depicts the mass center of the composite structure composed of blocks A , B , and C .

The main aim of our stability testing procedure in this paper is to get information about where to look for potentially unknown objects. If our stability algorithm is correct and we identify an unstable object, then we will look for an unknown supporting object. If we correctly detect an object as stable, we will not look for an unknown object. What is important is the impact of wrongly classifying an object as stable or unstable, and due to the impossibility of correctly evaluating stability of all situations, wrong stability results are unavoidable. So if we classify an object as stable even though it is unstable, we will not look for an unknown supporting object and might miss an object. The consequences of this might be negligible if we are running our algorithm on many similar images. Then there is a good chance that we identify the missed object at a later stage in a different image.

If we wrongly classify an object as unstable, we will check for an unknown supporting object. If we find none, then there is no difference between not looking for one and not finding one other than increased computation time. However, if we do find a supporting object, then this could either be a correct find, or it could be a fake object, for example background. Identifying a fake object can cause many problems, as it will be added to our list of known objects and will always be identified as an object. Therefore, we have to avoid identifying fake objects. The most important way to avoid fake objects is to be able to correctly identify background and to make sure that background can never be considered as an object (discussed in Section V-B).

The global stability algorithm improves the ability to identify objects as stable or unstable. Many situations that are incorrectly classified by the local stability algorithm can be corrected by our global stability algorithm. Fig. 4(a) gives an example of a locally stable structure that is detected as unstable by the global stability algorithm, and Fig. 4(b) demonstrates a situation that is locally unstable, but globally stable. In addition to the above two cases, our method can successfully deal with the configuration in Fig. 3 by evaluating the objects in terms of supporting relations rather than from top to bottom. Specifically, in the support graph, there is a path from A to B ; thus, when checking the stability of A , the mass center of A and B will be calculated and the support area will be determined corresponding to the direct supporters of both A and B .

```

1: procedure GLOBALSTABILITYTEST( $N_q$ ,  $SG$ )
2: Input:
3:  $N_q$  the node that refers to the querying object
4:  $SG$  a support graph that contains all objects in the structure including  $N_q$  as nodes and their support relations as edges
5: Output:
6: GlobalStability boolean value to state if the querying object is stable
7:
8:    $SupporterList \leftarrow \emptyset$ 
9:    $SupporteeList \leftarrow \emptyset$ 
10:  for all Node  $N$  in  $SG$  do
11:    if  $N$  supports  $N_q$  and  $SD(N, N_q) = 1$  then
12:       $SupporterList \leftarrow SupporterList \cup N$ 
13:    end if
14:    if  $N_q$  support  $N$  then
15:       $SupporteeList \leftarrow SupporteeList \cup N$ 
16:    end if
17:  end for
18:  if  $SupporterList = \emptyset$  then
19:    return false
20:  end if
21:  for all Node  $N$  in  $SupporteeList$  do
22:    for all Node  $N'$  in  $SG$  do
23:      if  $N'$  supports  $N$  and  $SD(N', N) = 1$  and  $N_q$  does not support  $N$  then
24:         $SupporterList \leftarrow SupporterList \cup N'$ 
25:      end if
26:    end for
27:  end for
28:  if vertical projection of the mass center of the substructure with nodes  $\in SupporteeList \cup N_q$  falls into the supporting area built with nodes  $\in SupporterList$  then
29:    return true
30:  else
31:    return false
32:  end if
33: end procedure

```

Fig. 6. Global stability test algorithm.

IV. IDENTIFYING UNKNOWN OBJECTS

We now present a general algorithm (see Fig. 7) that can automatically identify unknown objects in images obtained from 2-D video games. We assume that the computer vision method uses some form of image segmentation and clustering for detecting known objects. Which ones are used will depend on the application. In the following part, we use *Seg* and *Clus* to refer to an arbitrary segmentation and clustering algorithm respectively. Note *Seg* and *Clus* are not necessarily separate modules as segmentation could be done by clustering. The algorithm has four steps. Detecting known objects. The algorithm (see Fig. 7, lines 10–11) first uses *Seg* to obtain all the nonoverlapping regions in the image. *Clus* then classifies the regions to one of the known categories according to the selected visual features. A region will be labeled as *unknown* if there are no feasible categories. These are candidates for forming new objects.

Detecting locally unstable objects and possible areas of support. After detecting all the known objects, the algorithm (see Fig. 7, line 13) checks the local stability of each known object by setting the bottom line of the image as the initial ground. Once the algorithm detects a locally unstable object, it estimates the *support area* where possible supporters can be found. There are two possibilities that have to be checked for each locally unstable object (see Fig. 8): a) the area adjacent to the object that is underneath its center of mass; and b) the area adjacent to the object to the left of its center of mass up to its leftmost point (underneath the object and to its left), and the area adjacent to the object to the right of its center of mass up to its rightmost point (underneath the object and to its right). If there is already a known supporting object in one of those two areas we only

need to check the other area. If no locally unstable object is found, we go to Step 4).

Identifying new objects. The algorithm (see Fig. 7, lines 15–21) first checks area a) for a new object. This is an unknown region as defined above that is adjacent to the known object within the search area. The undetected object should be “touching” the unstable object, where touching does not necessarily mean that the objects actually connect, but there could be a small gap between them which depends on the application. If none is found, we check areas b) in the same way. If none is found, we skip this locally unstable object by marking it as locally stable, go back to Step 2) to the next unstable object. Once an adequate region is identified, the method will extract the features of the region and update the *Clus* with this information. The update can be a creation of a new category or a merge with one of the existing categories. Whenever *Clus* is updated, we go back to Step 1).

Detecting globally unstable objects. Once there are no locally unstable objects, the method (see Fig. 7, lines 25–34) will verify the global stability of all the identified objects using the global stability test algorithm (Fig. 6). If there is a globally unstable object, we estimate the searching area for unknown regions that can make the object globally stable. To do this, we take the object and all its supportees and consider it as a new substructure. Since the object is locally stable, it must be directly supported, either under its center of mass or to its left and its right. It can only be globally unstable, if the center of mass of the whole substructure falls outside the objects support base, either to its left or its right. If to its right, the area that needs to be checked for possible support is the area (underneath the substructure and to its right) that is directly adjacent to the whole substructure, on the right side of its support base up to its

```

1: procedure VISUAL DETECTION ALGORITHM( $\mathcal{I}$ , Seg, Clus)
2: Input:
3:  $\mathcal{I}$  a collection of images
4: Seg an image segmentation algorithm
5: Clus a clustering algorithm
6:
7: for all  $I$  in  $\mathcal{I}$  do
8:   repeat
9:     repeat
10:      Regions  $\leftarrow$  Seg( $I$ ) ▷ obtain non-overlapping regions by applying Seg to the image  $I$ 
11:      Unknown_Regions, Known_Objects  $\leftarrow$  Clus(Regions) ▷ use Clus to classify the regions
12:      for all Object  $O$  in Known_Objects do
13:        check the local stability of  $O$ 
14:        if  $O$  is locally unstable then
15:          estimate the support areas (local stability)
16:          if an unknown region is found in one of the areas then
17:            extract the features of the region, update Clus
18:            go to Line 10
19:          else
20:            mark  $O$  as locally stable, continue to check the next object in Known_Objects
21:          end if
22:        end if
23:      end for
24:    until There are no locally unstable objects
25:    for all Object  $O$  in Known_Objects do
26:      check the global stability of  $O$ 
27:      if  $O$  is not globally stable then
28:        estimate the support areas (global stability)
29:        if an unknown region is found in one of the areas then
30:          extract the features of the region, update Clus
31:          go to Line 10
32:        end if
33:      end if
34:    end for
35:  until There are no more globally unstable objects or there are no more unknown regions that can support the unstable objects
36: end for
37: end procedure

```

Fig. 7. Visual detection algorithm.

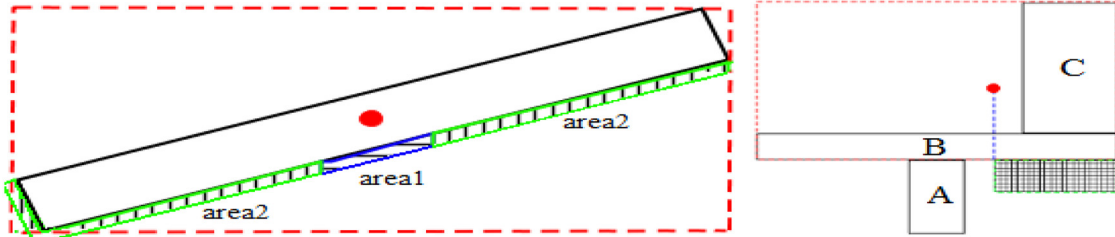


Fig. 8. (Left) Suggested searching areas for the locally unstable object. (Right) Suggested searching areas for the globally unstable object. The red dot indicates the center mass of B and C which is to the left of the supporting area provided by A.

rightmost point (see Fig. 8). If to its left, we need to check the similar area to the left of the support base. We now check the identified area for a new object in the same way as described in Step 3).

The method terminates when there are no more globally unstable objects or there are no more unknown regions that can support the unstable objects.

V. EXPERIMENTAL EVALUATION

First, we evaluate our method using the popular game *Angry Birds*. *Angry Birds* is a physics simulation game where an underlying physics simulator ensures that gravity and stability are enforced. If an object is unstable under gravity, it will fall until it is stable. In our evaluation, we only use images of stable situations, where all objects are individually stable and nothing moves. The images we use are the different levels available on the Chrome version of the game (chrome.angrybirds.com), in total 444 different levels. We use the initial configuration of

each level, which includes all objects that can possibly occur in these levels, and all of them are stable. Occasionally, objects appear to be floating in the air, so not all objects are always supported. There is an existing computer vision software (AI Birds 1.3) [12] that detects all relevant objects in the first 21 levels of the *Poached Eggs* series and identifies their category. The relevant object categories are hard-coded in the vision software, based on the primary colors that must be contained in these objects. In addition, the algorithm uses the Canny edge detection with settings that are manually optimized for detecting these objects. This algorithm can reliably detect the real shapes of all known objects, but is unable to detect any objects and object categories that are not explicitly hard-coded.

A. Application-Specific Vision Implementation

While we use a specific computer vision method to detect objects and object boundaries, other computer vision methods can be used instead. Which computer vision method is best

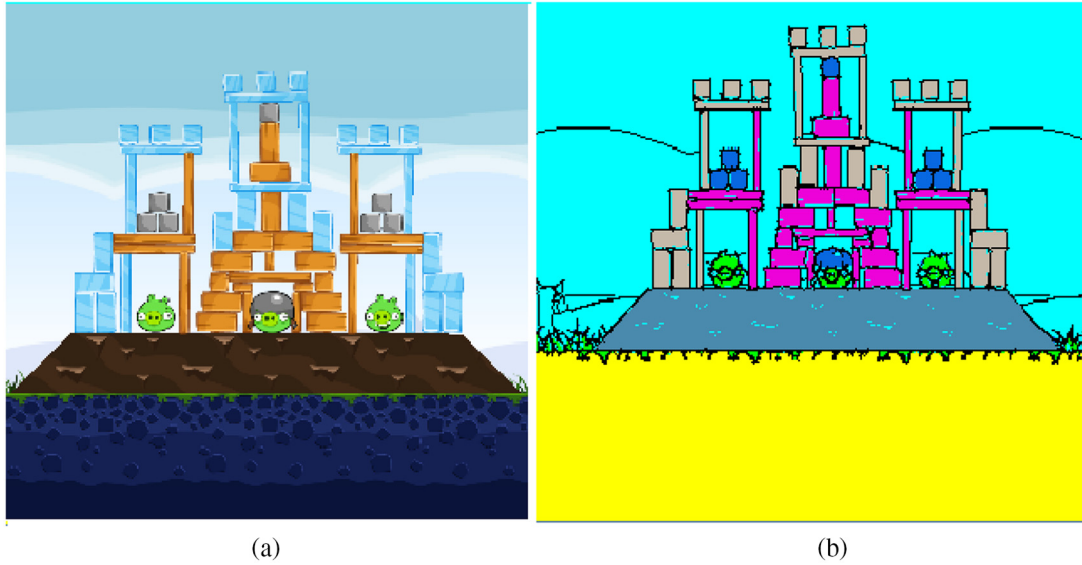


Fig. 9. (a) Typical POACHED EGGS level. (b) Detected objects. Objects of the same category have the same color.

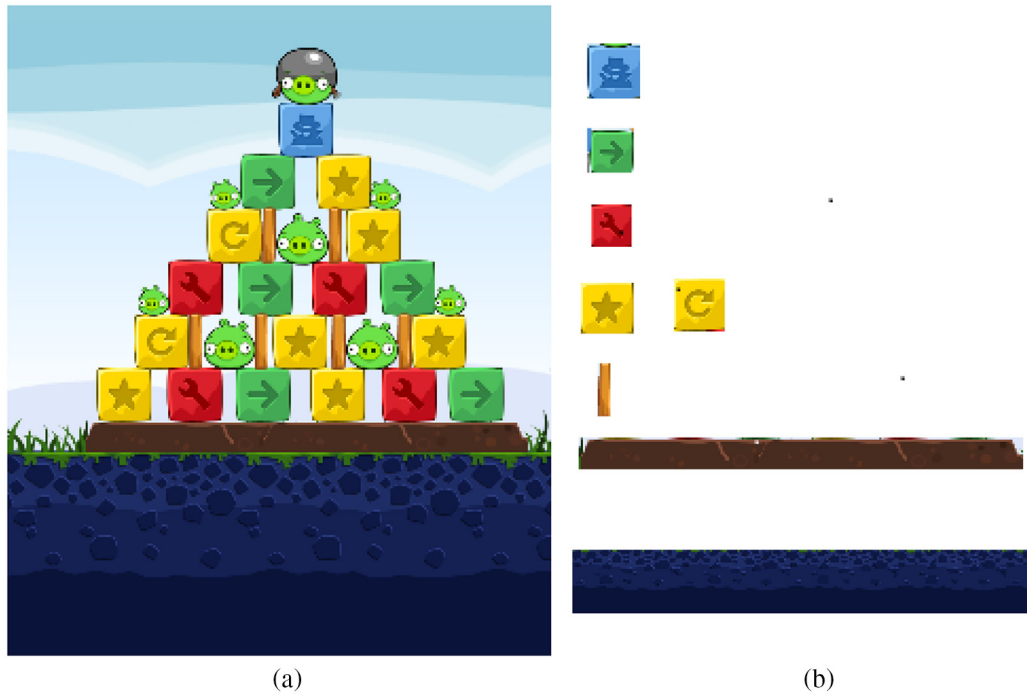


Fig. 10. (a) Level with many new blocks. (b) Newly detected objects; objects of the same category are arranged in the same row.

might depend on the particular application, can be adjusted accordingly, and is not important here. What is important, though, is our strategy of identifying unknown object using a qualitative stability analysis.

The specific computer vision method we use here distinguishes objects by colors (we use the *RGB* color space). Objects of similar colors will be identified as being of the same category. Our method maintains a list of entries for known object categories, with each entry describing the color distribution of a category. An entry contains a set of primary colors, the proportions of each color, and the total number of pixels that have been labeled by one of the colors.

We use a modified *K*-means [33], [34] algorithm for clustering. It has three parameters: the maximum number of clusters ($M_c = 10$), the minimum *RGB* color distance ($M_d = 40$) between the centroids of the clusters, and the maximum cluster radius ($M_r = 20$). The centroid of a cluster is computed by averaging the *RGB* values of the included pixels.

We use edge detection to assist with image segmentation. While many powerful segmentation algorithms [35], [36] have been developed recently, we use an early approach, the Canny edge detector [37], because its performance is sufficiently good to help us discover real shape of objects and then build correct support relations to detect potential objects. We apply



Fig. 11. (Left) A moon is present in the background. (Right) The wooden block with ID 2 is a core supporter of the stone block with ID 1 (remove the wooden block makes the stone block unstable). The moon segment with ID 3 is not a core supporter of the stone block (the stone will remain stable without the support of the moon segment).

the Canny edge detector with default settings to identify all edges in the image, and label each pixel with the most similar stored color. The similarity is measured by a weighted euclidean distance ΔC in $R'G'B'$, according to the formula $\|\Delta C\| = \sqrt{3 \times \Delta R'^2 + 4 \times \Delta G'^2 + 2 \times \Delta B'^2}$ [38]. Two colors are considered as similar if $\|\Delta C\| < M_r$. The method will label a pixel as unknown if there is no similar color in the entries. A connected component is a region enclosed by detected edges such that all pixels within that region have the same label. A connected component forms an object of a known category when most pixels within the component are labeled by the colors, with the same distribution, of the category.

Once a connected component is identified [in Step 3) of the visual detection algorithm], we identify its colors and the corresponding proportions. The method calculates the averages of the RGB values for the top three clusters by the size of the clusters. Those average RGB s are taken as primary colors of that component. The primary colors are further divided into three groups according to their proportions, namely *Dominant*: [40%, 100%], *Major*: [15%, 50%], *Minor*: (0%, 20%) ($[x, y]$ represents an interval between x and y , including both). A color can be in at most two adjacency groups. If the colors in the groups *Dominant* and *Major* are similar to the colors of the corresponding groups of any known category, the component will be classified to that category otherwise it will be identified as a new category. In the former case, the method will adjust the primary colors of the identified category by weighted average. For example, if one primary color rgb_x of the component is similar to rgb_y of the corresponding category, then the RGB value of the primary color will be updated by $(c_x * rgb_x + c_y * rgb_y) / (c_x + c_y)$ where c_x and c_y are the total number of pixels. The color proportion will be updated as well. In the latter case, the method will create an entry for the new object category.

B. Evaluation

In our first experiment, we ran our algorithm on the same 21 *Poached Eggs* levels. The only object we hard-coded is the green pig; all other objects cannot be detected initially. The only other game-specific setting we made is the definition of what is

classified as background, namely any object that occurs across the whole width of the image, not necessarily in one piece.

Our algorithm detected 1194 new objects in these 21 levels. We clustered these objects according to their primary colors and their shapes. We used two different clustering methods. The first method we used is the method described in Section IV, where objects are clustered based on the primary colors of group *Dominant* and *Major* (2C). In the second method (1C), we classified objects according to their most prominent colors by considering *Dominant* only. Other methods could be used as well. For 1C, we obtained seven different categories of objects, and for 2C we obtained 12 different categories. The object shapes formed subcategories of these categories.

The object categories identified by 1C correspond exactly to the hard-coded object categories of the AI Birds 1.3 vision system. Fig. 9 shows the object classifications made by clustering method 1C after running on the 21 levels. We can detect all the objects detected by the AI Birds 1.3 vision system and, in all of the cases, the categories that are the same for AI Birds 1.3 are also the same in our algorithm. On average, our algorithm took 4 s per level and we did three iterations until we received stable detections (that is after the second iteration no new object categories were identified).

Next we applied our algorithm to all 444 available levels. We loaded them in random order. Fig. 10 shows an example where many building blocks are different from those in the *Poached Eggs* levels. Our algorithm detected all the building blocks while the AI Birds 1.3 vision system can only detect the pig and wood blocks. Similar to this example, the 444 levels have a number of new objects and many different background settings, some of them with very complicated background structures. Some levels were particularly difficult as the background contained similar colors to already detected objects, or it contained objects that did not satisfy our aforementioned criterion of occurring across the whole width of the image. For example, one type of background in the *treat or trick* episodes contains a moon which is right behind the object structure, and does not occur anywhere else in the background (see Fig. 11). The moon can be detected when it is located in the support area of some unstable objects. We deal with such background objects by examining the probability of the object being a core supporter

TABLE I
APPEARANCE AND CORE SUPPORT NUMBERS OF SOME TYPICAL
OBJECTS IN 200 RANDOM LEVELS #APPEAR: TOTAL NUMBER
OF DETECTED OBJECTS, #CORESUPP: NUMBER OF TIMES
OBJECTS ARE THE CORE SUPPORTERS OF OTHER
OBJECTS. C/A: #CORESUPP / #APPEAR

Object Category	#Appear	#CoreSupp	C/A
Stone	3278	2423	0.65
Wood	4060	2849	0.70
Moon (Background)	30	2	0.07
Brown Hill	428	1224	2.86

of other objects. A core supporter c of a stable object s is a supporter that makes s unstable if c is removed. In most cases, a stable object (not floating) “supported” by a background object is actually supported by other foreground objects. Once those foreground objects are detected, the background object will no longer be the core supporter of the object. By making use of this knowledge, the algorithm filtered most of such background objects. As Table I shows, the object category *Moon* appeared 30 times but was core supporter only twice. In contrast, objects of category *Brown Hill* were 1224 times core supporters with only 428 appearances.

We evaluate the method using metrics of precision and recall

$$\text{recall} = \frac{|\{\text{detected categories}\} \cap \{\text{actual categories}\}|}{|\{\text{actual categories}\}|}$$

$$\text{precision} = \frac{|\{\text{detected categories}\} \cap \{\text{actual categories}\}|}{|\{\text{detected categories}\}|}.$$

The set *actual categories* comprises all the categories of foreground objects (which support other objects). We identified 38 categories in the game. As a result of evaluating our method on all 444 levels, we obtained 25 categories (recall = 0.45, precision = 0.68) with 1C and 54 categories (recall = 0.79, precision = 0.55) with 2C. Fig. 12 shows some of the detected objects through this procedure.

While most of the categories distinguished by 1C are meaningful distinctions, some objects that should form different categories were clustered together. Those were distinguished by 2C, which also distinguished object categories that should be in the same category, for example, some ice blocks (the blue blocks shown in Fig. 9) are translucent, and their color will change as background changes.

In our final experiment, we applied our algorithm to two other 2D video games, *Candy Crush* (www.candycrushsaga.com) and *Super Stacker* (www.superstacker.com) where a downward gravity is enforced. Our algorithm started with the green candy (or yellow face in super stacker) as the only known object, and it detected all the other objects (see Figs. 13 and 14).

It is clear that by optimizing the clustering method we can achieve more accurate object categories. However, the actual object categories strongly depend on the particular application we are using and clearly cannot be generalized across different applications. Additional methods such as supervised learning could be used to optimize the resulting categories for

a particular application. More important for the scope of our paper is that our evaluation confirms that in 2-D physics-based video games unknown objects can be reliably detected based on a stability analysis of already known objects.

VI. DISCUSSION

Being able to see, identify, and recognize objects visually is an essential requirement for any AI agent interacting with the physical world. The current state of the art in computer vision limits this ability and allows reliable recognition of objects mainly for previously learned objects. We propose an approach for combining computer vision methods with methods from qualitative spatial reasoning, to improve detection of unknown objects. Our approach relies on the fact that objects typically do not float in the air, but require physical support to keep them stable. Once an object is detected that appears to be unsupported, there is strong evidence that there must be a yet undetected object that supports it. We use a method developed in the area of qualitative spatial reasoning to infer whether already detected objects are stable or unstable and to infer where a possible support would be located if detected unstable. Standard computer vision methods can then be used to identify potential objects in the inferred location, and standard clustering methods to identify the category of newly identified objects.

Since our approach heavily relies on the performance of underlying computer vision methods, we evaluated it in some 2-D video games where known objects can be reliably detected and where other open computer vision problems do not affect the performance of our method. It turns out that our method can accurately detect relevant objects and cluster them into relevant object categories. It achieves the same accuracy and reliability as existing vision software where all detected object categories are hard-coded. In addition, we can detect and categorize many new objects that the existing software fails to detect.

This example provides an important proof of concept that our method is useful for detecting unknown objects. There are several directions for future work in order to improve object clustering and the accuracy of the stability analysis. 1) We can use supervised learning methods to optimize the clustering of objects into object categories relevant for a given application domain. 2) The current method is a passive method that is only based on analyzing images. Actively interacting with the observed environment can further improve the performance. In the case of *Angry Birds*, we could shoot a bird at each of the detected objects and analyze how the object reacts. This would help in clustering objects, but also in obtaining physical properties of object categories. The lack of information about the physical properties of the observed object is also a reason why the stability analysis we perform is only an approximation. By interacting with the world we could obtain information about the actual mass of objects which would improve the stability calculation. Once game objects can be detected automatically, this opens up a whole range of possible new research and applications. This goes from building general game playing agents [24], [39], [40] that can play physics-based video games to creating intelligent nonplayer characters (NPCs) that are able to recognize unseen objects.



Fig. 12. Some of the detected objects in the 444 levels.

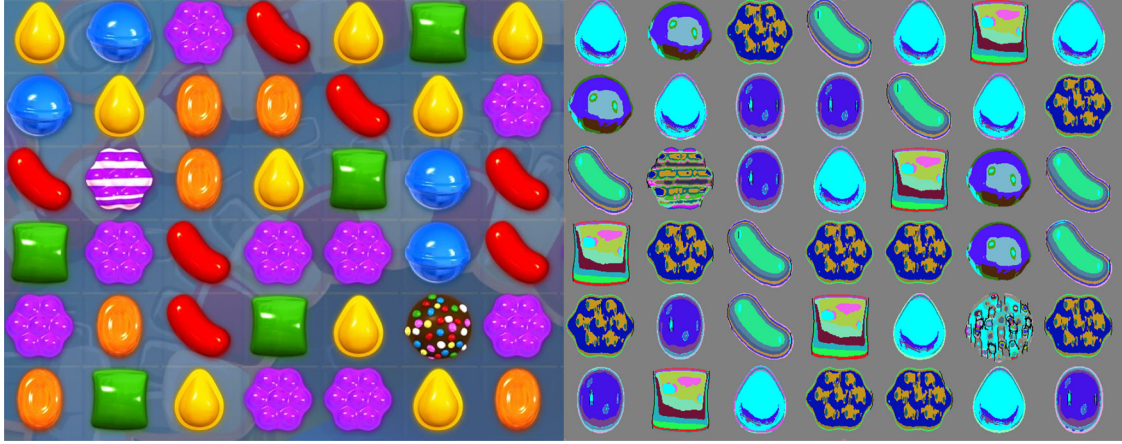


Fig. 13. (Left) The image of a *Candy Crush* level. (Right) Image segmentation (objects of the same category has the same color composition) after the algorithm detected all the objects.

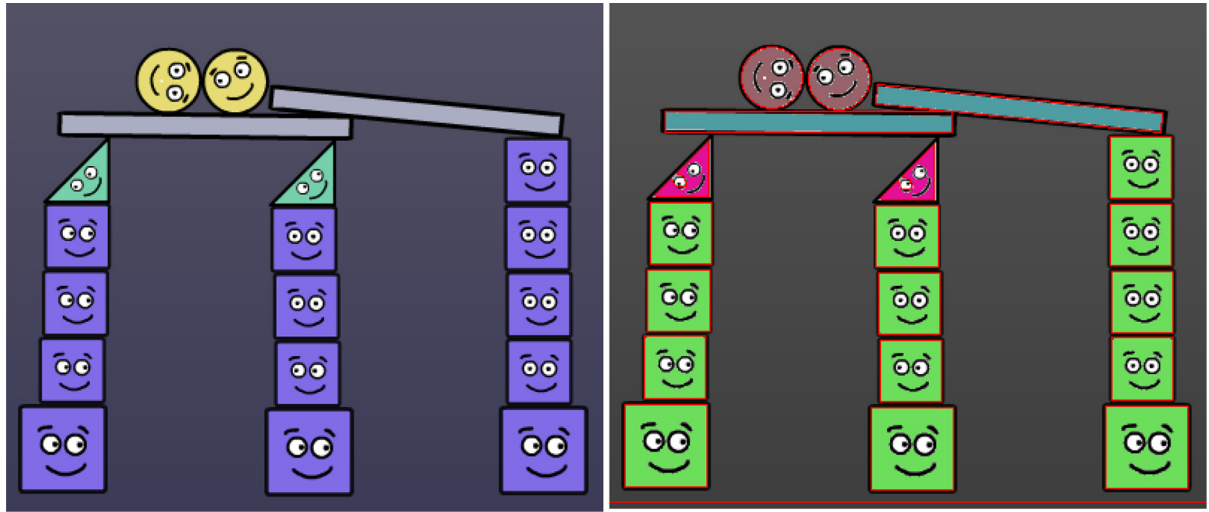


Fig. 14. (Left) The image of a super stacker level. (Right) Image segmentation (objects of the same category are shown in the same color) after the algorithm detected all the objects.

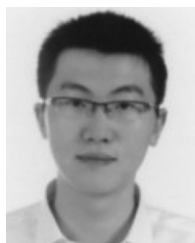
The learning-based agents in Atari-GGP would also benefit from this method. Since the method could be generally applied to detect objects in 2-D games where the gravity is present, learning agents that perform object detection could use this method to complement their current vision modules. For example, Hausknecht *et al.* [23] could use this method as a replacement of the manual object detection module.

In addition, it is possible to adapt the method to a 3-D setting. The current rules for the stability analysis extend

naturally to the third dimension. The information about the third dimension (depth) could be directly extracted from *RGBD* images. Therefore, the method has the potential of being applied to indoor scene understanding. For example, it could complement the method in [41] on estimating the support relations. To be able to successfully apply the method in a real-world setting, one has to devise new rules to cope with new stable scenarios (e.g., a lamp hanging from the ceiling).

REFERENCES

- [1] P. M. Roth and M. Winter, "Survey of appearance-based methods for object recognition," *Inst. Comput. Graph. Vis., Graz Univ. of Technology Graz, Austria: Tech. Rep.*, 2008.
- [2] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.
- [3] C. H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 951–958.
- [4] Y. J. Lee and K. Grauman, "Object-graphs for context-aware visual category discovery," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 2, pp. 346–358, Feb. 2012.
- [5] M. Genesereth, N. Love, and B. Pell, "General game playing: Overview of the AAAI competition," *AI Mag.*, vol. 26, no. 2, p. 62, 2005.
- [6] J. Levine *et al.*, "General video game playing," *Artif. Comput. Intell. Games*, vol. 6, pp. 77–83, 2013.
- [7] J. Renz, X. Ge, S. Gould, and P. Zhang, "The Angry Birds AI Competition," *AI Mag.*, vol. 36, no. 2, pp. 85–87, 2015.
- [8] K. Grauman and B. Leibe, *Visual Object Recognition*, New York, NY, USA: Morgan & Claypool, 2011.
- [9] J. Matas and S. Obdrzalek, "Object recognition methods based on transformation covariant features," in *Proc. Eur. Signal Process. Conf.*, 2004, pp. 1721–1728.
- [10] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 4, pp. 509–522, Apr. 2002.
- [11] R. Jain, R. Kasturi, and B. G. Schunck, *Machine Vision*, New York, NY, USA: McGraw-Hill, 1995, vol. 5.
- [12] X. Ge *et al.*, *Angry Birds Game Playing Software Version 1.3: Basic Game Playing Software*, 2014, <http://www.aibirrds.org/basic-game-playing-software.html>.
- [13] J. Hosang, R. Benenson, P. Dollár, and B. Schiele, "What makes for effective detection proposals?," 2015, <http://arxiv.org/abs/1502.05082>.
- [14] C. H. Lampert, M. B. Blaschko, and T. Hofmann, "Efficient subwindow search: A branch and bound framework for object localization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2129–2142, Dec. 2009.
- [15] N. Loeff, H. Arora, A. Sorokin, and D. Forsyth, "Efficient unsupervised learning for localization and detection in object categories," *Advances in Neural Information Processing Systems*, Cambridge, MA, USA: MIT Press, 2005, pp. 811–818.
- [16] V. Hedau, D. Hoiem, and D. Forsyth, "Recovering free space of indoor scenes from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 2807–2814.
- [17] S. Branson *et al.*, "Visual recognition with humans in the loop," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 438–451.
- [18] K. Grauman, and T. Darrell, "Unsupervised learning of categories from sets of partially matching image features," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2006, vol. 1, pp. 19–25.
- [19] Y. J. Lee and K. Grauman, "Learning the easy things first: Self-paced visual category discovery," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 1721–1728.
- [20] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *J. Artif. Intell. Res.*, vol. 47, no. 1, pp. 253–279, 2013.
- [21] V. Mnih *et al.*, "Playing Atari with deep reinforcement learning," 2013, <http://arxiv.org/abs/1312.5602>.
- [22] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [23] M. Hausknecht, J. Lehman, R. Miikkulainen, and P. Stone, "A neuroevolution approach to general Atari game playing," *IEEE Trans. Comput. Intell. AI Games*, vol. 6, no. 4, pp. 355–366, Dec. 2014.
- [24] M. Hausknecht, P. Khandelwal, R. Miikkulainen, and P. Stone, "Hyperneat-GGP: A hyperneat-based Atari general game player," in *Proc. 14th Annu. Conf. Genetic Evol. Comput.*, 2012, pp. 217–224.
- [25] M. Blum, A. Griffith, and B. Neumann, "A stability test for configurations of blocks," *Rep. AIM-188*, 1970.
- [26] P. Zhang and J. Renz, "Qualitative spatial representation and reasoning in Angry Birds: The extended rectangle algebra," in *Proc. 14th Int. Conf.: Principles Knowl. Representation Reason.*, 2014, <http://www.aaai.org/ocs/index.php/KR/KR14/paper/view/8021>.
- [27] A. Mukerjee, and G. Joe, "A qualitative model for space," in *Proc. AAAI*, 1990, pp. 721–727.
- [28] P. Balbiani, J. Condotta, and L. del Cerro, "A new tractable subclass of the rectangle algebra," in *Proc. Int. Joint Conf. Artif. Intell.*, 1999, pp. 442–447.
- [29] A. Gupta, A. A. Efros, and M. Hebert, "Blocks world revisited: Image understanding using qualitative geometry and mechanics," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 482–496.
- [30] B. Zheng, Y. Zhao, J. C. Yu, K. Ikeuchi, and S.-C. Zhu, "Beyond point clouds: Scene understanding by reasoning geometry and physics," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 3127–3134.
- [31] Z. Jia, A. Gallagher, A. Saxena, and T. Chen, "3D-based reasoning with blocks, support, and stability," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, DOI: 10.1109/CVPR.2013.8.
- [32] P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum, "Simulation as an engine of physical scene understanding," in *Proc. Nat. Acad. Sci.*, vol. 110, no. 45, pp. 18327–18332, 2013.
- [33] Y. Gong, G. Proietti, and C. Faloutsos, "Image indexing and retrieval based on human perceptual color clustering," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 1998, pp. 578–583.
- [34] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Stat. Probab.*, 1967, vol. 1, no. 14, pp. 281–297.
- [35] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *Int. J. Comput. Vis.*, vol. 59, no. 2, pp. 167–181, 2004.
- [36] Y. Boykov and G. Funka-Lea, "Graph cuts and efficient ND image segmentation," *Int. J. Comput. Vis.*, vol. 70, no. 2, pp. 109–131, 2006.
- [37] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, PAMI-8 no. 6, pp. 679–698, Jun. 1986.
- [38] C. Poynton, "Frequently asked questions about color," 1997, <http://www.poynton.com/ColorFAQ.html>.
- [39] H. Finnsson, and Y. Björnsson, "Simulation-based approach to general game playing," in *Proc. 23rd Nat. Conf. Artif. Intell.*, 2008, vol. 8, pp. 259–264.
- [40] T. Cerexhe, D. Rajaratnam, A. Saffidine, and M. Thielscher, "A systematic solution to the (de-)composition problem in general game playing," in *Proc. Eur. Conf. Artif. Intell.*, 2014, pp. 195–200.
- [41] P. K. Nathan Silberman, D. Hoiem, and R. Fergus, "Indoor segmentation and support inference from RGBD images," *Computer Vision-ECCV 2012, Lecture Notes in Computer Science Berlin, Germany: Springer-Verlag*, 2012, vol. 7576, pp. 746–760.



Xiaoyu Ge received the B.S. degree in information technology from Monash University, Melbourne, Vic., Australia, in 2011 and the B.S. honors degree in artificial intelligence from the Australian National University, Canberra, A.C.T. Australia, in 2012, where he is currently working toward the Ph.D. degree.

His research interests include physics reasoning, qualitative spatio-temporal reasoning, and developing AI that can safely and successfully interact with the physical world.



Jochen Renz received the Ph.D. degree in artificial intelligence from the University of Freiburg, Freiburg im Breisgau, Germany, in 2000 and the Habilitation degree in information systems from the Vienna University of Technology (Technische Universität Wien), Vienna, Austria, in 2003.

He is a Professor in the Research School of Computer Science, Australian National University (ANU), Canberra, A.C.T. Australia and the Head of the ANU Artificial Intelligence Group. His main research interests are in qualitative spatial reasoning,

both its theoretical foundations and its applications.



Peng Zhang received the B.E. degree in intelligence science and technology from Beijing University of Posts and Telecommunications, Beijing, China, in 2011 and the M.C. degree in artificial intelligence from the Australian National University, Canberra, A.C.T. Australia, in 2013, where he is currently working toward the Ph.D. degree under the supervision of Prof. J. Renz in the Research School of Computing Science.

His research interest is qualitative spatial representation and reasoning.