Multiple Object Tracking using Spatial Reasoning

Abstract

Intelligent agents perceive the world mainly through images captured at different time points. Being able to track objects from one image to another is fundamental for understanding the changes of the world. Humans can perform efficient object tracking by common sense reasoning. For example, we know that a free falling object will not change its downwards movement unless acted upon by another force. Given an initial image with a free falling object, we will first check the objects which are below the initial object location in order to find the corresponding object in subsequent images.

Object tracking has been extensively studied within the context of computer vision. However, reasoning is typically not used to solve this problem. Most tracking algorithms represent and track objects by their visual features and trajectories. Those algorithms become error prone when there are multiple objects with the same appearance that follow similar trajectories, that change trajectories, or when trajectories are not available. Our solution is closer to human reasoning in that it considers spatial relations and physical properties of objects. It can improve tracking accuracy significantly by qualitatively predicting possible motions of objects and discarding matches that violate spatial constraints.

We evaluate the solution in a real video gaming scenario. We capture sequences of screenshots of the game where multiple objects are moving, and compare the tracking accuracy by varying the length of the intervals between the time points at which the screenshots are taken. Our solution is flexible enough that can be extended with more powerful qualitative reasoning models to tackle multiple object tracking when the time interval between images is long.

Introduction

Visual perception is one of the main sources of information about the world. Most intelligent agents have optical devices to "see" their environment and to detect changes. Image understanding (Sonka et al. 1999; Sridhar, Cohn, and Hogg 2011) and object detection (Papageorgiou, Oren, and Poggio 1998) are essential methods for extracting useful information from images. Equally essential is object tracking (Yilmaz, Javed, and Shah 2006), the ability to identify the same object in a series of images or in video and to track its movement and changes. Existing object tracking methods typically rely on the visual appearance of objects and on their trajectories to successfully track objects (Yilmaz, Li, and Shah 2004; Cutler and Davis 2000; Viola, Jones, and Snow 2005).

In this paper we look at the problem of tracking perceptually indistinguishable objects (PIO) (Santore and Shapiro 2005), i.e., objects in images or video that have the same appearance and cannot be distinguished. We want to be able to identify which PIO at time t_2 is identical to which PIO at time t_1 without continuously monitoring the changes between t_1 and t_2. Under the assumption of discrete observations we have to be able to reidentify each PIO whenever we obtain a new observation (Pollock 1974). While all permutations of identity assignments are theoretically possible, the task is to find an assignment that is consistent with a physical event that is responsible for the changes. One example of this problem is the game of Snooker where there are 15 indistinguishable red balls. After every shot the locations of some balls change and we want to infer which red ball after a shot is identical to which red ball before a shot, such that the changes can be explained by a single cue strike. Solving this problem requires us to estimate the physical effects that caused the changes.Another example is bowling where we strike indistinguishable pins with a bowling ball.

Our interest in this problem is motivated by the Angry Birds AI competition (AIBirds 2013) where the task is to build an AI agent that can play the popular game Angry Birds as good as the best human players. A major problem in this context is to accurately predict the outcome of a shot, i.e., to infer how the game objects move when hit by a bird in a particular way. One way of estimating consequences of shots is to know which object after a shot corresponds to which object before a shot. Then we can use this information to learn consequences of actions by using before and after object locations as input to machine learning algorithms. Once we can estimate consequences of shots, it becomes possible to plan good shot sequences that can solve given game levels. Therefore, identifying which objects after a shot correspond to which objects before a shot is an important step in building a sophisticated Angry Birds AI agent.

We develop an algorithm that allows us to infer identity assignments of PIOs that are consistent with the physical effects of a single impact. We evaluate our proposed solution using the Angry Birds scenario. We take subsequent screenshots of an active Angry Birds game taken with varying time gaps and apply our method to match the objects be-

tween successive screenshots. We measure the accuracy of our method by using the percentage of correct matches out of the total number of possible mismatches. As expected, it turns out that the smaller the time gaps, the higher the accuracy of the matches. But overall the quality of our matches is very high.

Detection and Representation of Objects in Images

In order to be able to track objects in images, we obviously have to be able to first obtain objects from images. Object recognition (Belongie, Malik, and Puzicha 2002; Lowe 1999) is one of the hardest and most researched problems in Computer Science and it is still unsolved in its generality. In this paper we assume that objects can be detected and we will use cases where object detection is solved and works. In particular, we use images taken from the Angry Birds game, as this is the main motivation of our work in this paper. The basic software provided by the Angry Birds AI competition organisers includes an object recognition module that detects all known objects with reasonable accuracy and places a minimum bounding rectangle (MBR) around them. In this paper we use an advanced version of the object recognition module which is based on a student project by Andrew Wang and which will be integrated to the basic software soon (Wang 2013). This software detects the real shapes of all known objects plus the contour of the ground. For our paper, we only consider rectangular objects and circular objects. We use an MBR to approximate the region occupied by a circle and use exact shapes for the general solid rectangles (GSR), i.e. rectangles that can have any angle and are impenetrable, as is the case in Angry Birds.

To represent these objects we use a qualitative spatial representation in addition to the real shape and location of the objects. Many rectangle-based qualitative spatial calculi (Balbiani, Condotta, and Del Cerro 1998; Cohn, Renz, and Sridhar 2012; Sokeh, Gould, and Renz 2013) have developed in the context of Qualitative Spatial Reasoning (Cohn and Renz 2008). These calculi typically deal with one or more spatial aspects such as topology, size, direction, or distance, and make a number of qualitative distinctions according to these aspects. There is currently only one spatial calculus that specifically deals with rectangles of arbitrary angles, the GSR-n calculus proposed by (Ge and Renz 2013). GSR-n is a comprehensive spatial representation of GSRs. It defines eight contact sectors that correspond to the eight edges and corners of the rectangles. As shown in Figure 1(a) and (b), we distinguish eight sectors for regular rectangles $(R_1, \ldots, R_8)$ and eight sectors for angular rectangles $(A_1, \ldots, A_8)$. Given two GSRs o_1 and o_2 that contact each other via $sector_1, sector_2 \in \{A_1, ..., A_8, R_1, ..., R_8\}$, the contact relation between o_1 and o_2 can be expressed as the constraint $o_1\ (sector_1, sector_2)\ o_2$ (Figure 1.c). With the contact relations, GSR-n allows us to distinguish if and how two objects contact each other. Since the objects can only interact via contacts, we can further infer the possible motions of an object from the GSR relations the object holds with others. GSR-n also defines a set n of unary relations that qualitatively describes the size and leaning direction of a GSR. Since we represent the rectangles using their real shapes, we do not need the unary relations.

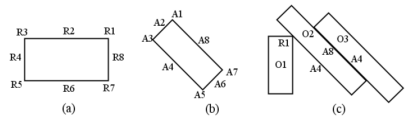

Figure 1: *Contact sectors of (a) a normal rectangle (without rotation) and (b) an angular rectangle. (c) An example scenario where* $o_1(R_1, A_4)o_2$, $o_2(A_8, A_4)o_3$

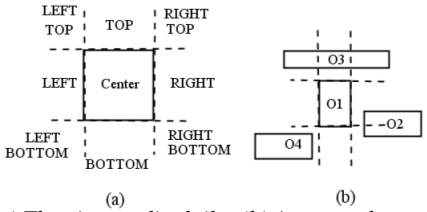

Figure 2: *(a) The nine cardinal tiles (b) An example scenario where* o_3 (TOP) o_1, o_2 (RIGHT) o_1, *and* o_4 (BOTTOM LEFT) o_1

Objects Representation with the extended GSR-n relations (EGSR)

Given two GSRs, we obtain the contact relation by enumerating all the plausible combinations of the two GSRs' contact sectors and for each combination calculating the distance between the two sectors. The combination with the shortest distance constitutes the contact relation. Note, the shortest distance can be non-zero. A non-zero distance means the two GSRs are separate, otherwise touch. So unlike the original definition that the contact relation can only be assigned when two GSRs touch, here we can assign a contact relation for any pair of GSRs. To distinguish whether two GSRs touch or not, we turn to qualitative distance relations which are covered in the next section.

The problem with GSR-n is that it uses $(\emptyset, \emptyset)$ to represent the spatial relation between all the non-touched GSRs. Thus, it introduces ambiguities when the rectangles are disconnected. To overcome the ambiguities, we extend the original GSR-n by integrating it with the cardinal tiles (Goyal and Egenhofer 1997). Specifically, we partition the embedding space around an reference object into nine mutually exclusive tiles(Figure 2.a). The centre tile corresponds to the minimum bounding rectangle of the object and the other eight tiles correspond to the eight cardinal directions. We call the *LEFT, RIGHT, BOTTOM, TOP* core tiles.

The new representation is called EGSR (Figure 3). Given a set $\mathcal{B}_{GSR}$ of GSR contact relations and a set $\mathcal{B}_{card}$ of cardinal tiles, we add $\perp$ to both sets to indicate a unassigned relation, an EGSR relation is then written as $(r_1, r_2), r_1 \in \mathcal{B}_{GSR} \cup \{\perp\}, r_2 \in \mathcal{B}_{card} \cup \{\perp\}$. We abbreviate (r_1, r_2) by the cardinal tile r_2 or by the contact relation r_1 if it is clear which one is meant.

We compute the EGSR relation between two spatial objects by first checking whether their MBRs intersect or boundary touch. If so, we assign a GSR-n relation accordingly. Otherwise, one of the eight cardinal tiles will be used. If one object's MBR occupies multiple tiles of the referred object, we will assign the core tile occupied by the MBR (Figure 2.b). Note, when one object's MBR occupies more than one core tiles of the referred object, it must intersect or boundary touch the referred object's MBR, in this case, GSR-n relation will be assigned instead. All EGSR relations are obviously converse, e.g. the converse of *TOP LEFT* is *BOTTOM RIGHT*, the converse of (R_4, S_7) is (S_7, R_4).

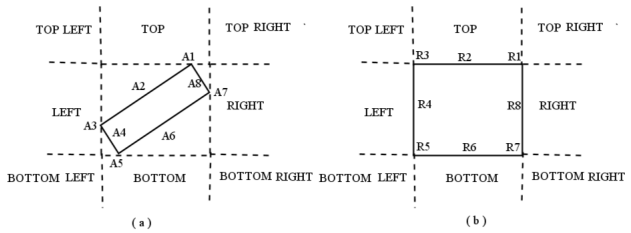

Figure 3: *Contact sectors and Cardinal directions of (a) an angular rectangle and (b) a normal rectangle*

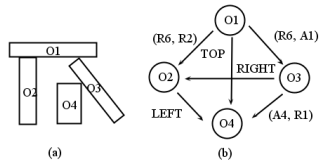

Figure 4: *(a) A spatial scenario where the four rectangles form a stable structure under downward gravity (b) The corresponding qualitative constraint network*

(Wallgrün, Wolter, and Richter 2010) introduced the notion of qualitative interpretation that extracts qualitative descriptions from the input data. A scenario containing multiple spatial objects can be qualitatively interpreted by EGSR via a qualitative constraint network (QCN). QCN is a labelled graph where each node corresponds to an object and directed edges represents relational constraint that has to hold between the two objects. Figure 4 shows an example of a QCN based on EGSR relations.

Efficient Matching by Approximating Movement

The search space of the problem is huge as there is a combinatorial number of potential matches between any two object sets. However, most matches can be avoided by searching through the corresponding objects only in a limited area. The area of the initial object should therefore cover all the objects in the subsequent scene that can be potentially matched to the initial object. We use a circular region to represent this area. The circle's centre is located at the centroid of the initial object and the radius of the circle is the maximum shift of the centroid. The radius is calculated as $v \times t$ where v is the maximum velocity of the object and t is the time gap between the initial and subsequent scenes. This calculation ensures that the circle can adapt to different time gaps. We call this circle the *movement bounding circle* (MBC).

The relative distance between two objects can then be allocated to three meaningful classes, namely touch, reachable, and non-reachable (Figure 5.a. An object o can touch another object o' at one of its contact sectors. o' is reachable by o if o' is within the MBC of o otherwise non-reachable.

The MBC can be divided into four quadrants to further restrict the search area. A quadrant of an object is said to be active if the object is likely to be in that quadrant at the next time point, otherwise the quadrant is inactive. The search space can be reduced by first searching possible matches in the active quadrants, if there are no such matches, then searching in the other quadrants. Given a MBC C, the active quadrants are $C^{(i,j)}, i, j \in \{-, +, *\}$ where (+,+), (+,-), (-,-), (-,+) correspond to the top-right, top-left, bottom-left, bottom-right quadrants respectively (Figure 5.b). (*, *) refers to an arbitrary quadrant.

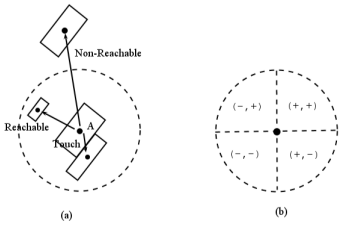

Figure 5: *(a) Qualitative distance with respect to object A and its MBC (b) The four quadrants of a MBC*

We can infer the active quadrants for an object by approximating the movement direction of the object, i.e. by estimating which of the quadrants the object is most likely to be in at the next time point. Object movement can be inferred from impact. Knowing the direction and the force of an impact, one can approximate the subsequent movements of the objects affected by the impact, directly or indirectly. When the impact information is not available, we can still approximate the movement by analysing structural properties, e.g. the stability of an object or a group of objects. An object is stable when it is supported and remains static.

We call the movement approximating problem we want to solve MQSR(Θ, R, $M_{\mathcal{R}}$) where Θ is a set of objects in a spatial scenario and R is a qualitative spatial representation of the objects. $M_{\mathcal{R}}$ is a reasoning model composed of a set of configurations written in the relations defined by R by satisfying which the corresponding active quadrants of an object can be obtained. For each object in Θ, we want to estimate the object's active quadrants by evaluating the object's spatial configurations against $M_{\mathcal{R}}$.

(Ge and Renz 2013) defined four kinds of supports that can make a solid rectangle stable and provided the corresponding GSR configurations. Our model $M_{\mathcal{R}}$ contains all such stable configurations. Given each object, the model will evaluate the stability of the object by the configurations and return $C^{(*,-)}$ for the unstable objects. We solve the problem in the Angry Birds scenario where a bird hit usually comes from the left. We use EGSR to represent the spatial configuration of a scenario. The model we use is mainly based on the stability analysis. A stable object may become unstable if it loses a support due to a bird hit. From the bird's trajectory, we can determine which object will be hit by the bird, and approximate the stability of the resulting scenario with the removal of that object (Figure 6).

We can get a more restricted area, $C^{(+,-)}$ or $C^{(-,-)}$, by analysing the direction in which the object is falling. For example, a right / left leaning rectangle will fall to the right / left if there is no support at the right / left side, and the corresponding active quadrant is $C^{(+,-)}$ / $C^{(-,-)}$. Here we illustrate how can we express the configurations described in the example using EGSR relations. We denote the left leaning and right leaning objects as O^L, O^R respectively, and the MBC of an object o is written as C_o. The two configuration can be expressed using EGSR relations:

1. Right-leaning: $\forall o_1^R : \exists o_2^* : o_1^R(A_5, *)o_2^* \wedge \neg\exists o_3^* : o_1^R(A_6, *)o_3^* \wedge \neg\exists o_4^* : o_1^R(A_7, *)o_4^* \Rightarrow C^{(+,-)}_{o_1^R}$
2. Left-leaning: $\forall o_1^L : \exists o_2^* : o_1^L(A_5, *)o_2^* \wedge \neg\exists o_3^* : o_1^L(A_3, *)o_3^* \wedge \neg\exists o_4^* : o_1^L(A_4, *)o_4^* \Rightarrow C^{(-,-)}_{o_1^L}$

The model $M_{\mathcal{R}}$ contains all such configurations and can be

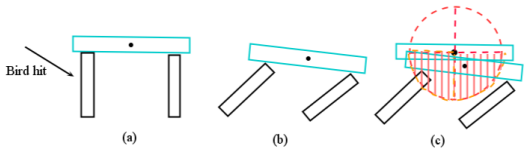

Figure 6: *(a) The object in cyan is stable, and the impending impact is indicated by the arrow (b) A subsequent scene after the bird hit (c) The estimated active quadrant (shadowed area)*

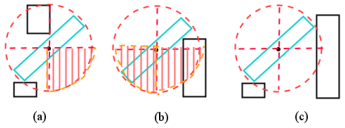

Figure 7: *The active quadrants of o_1 is (a) $C^{(+,-)}$ by satisfying the Right-leaning configuration (b) $C^{(*,-)}$ because o_1 is unstable and there is no object contacts A_5 of o_1 (c) No active quadrants as the object is stable*

replaced with any other model using EGSR relations. Figure 7 shows some example scenarios and the corresponding quadrants estimated by $M_{\mathcal{R}}$. We will show the model $M_{\mathcal{R}}$, although simple, is sufficient to solve the problem in the evaluation part.

Handling Common Movement by Spatial Reasoning

One challenge is to determine a match between PIOs that are close to each other and have similar trajectories. Figure 8.a shows a scene where objects A and B form a slope and three indistinguishable squares, o_1, o_2 and o_3, are lying on the slope. Figure 8.b is a subsequent image where the three squares have rolled down slightly. There are 6 ways in total to match the squares in the two images but only $\{o_1 \sim o_4, o_2 \sim o_5, o_3 \sim o_6\}$ is possible ($\sim$ is a operator that matches one object to another). Because they do not use reasoning about the spatial relations between the squares, some optimization algorithms, e.g. minimizing centroid shift, will tend to match o_2 with o_6. In many cases, object dynamics, such as velocities, are unavailable, thus many of the state-of-the-art tracking algorithms will not work well because of the lack of a suitable dynamic model.

Humans can solve this case efficiently using spatial reasoning. Since we know the objects are moving at a similar velocity, the relative spatial changes among them are subtle. Hence the spatial relations between those objects are unlikely to become converse while they are moving. When matching, we try to keep the original spatial relations among the subsequent objects. We emulate this reasoning in testing a match by first identifying those objects that are following a similar trajectory and then determining whether any relation has become converse in the subsequent image.

Objects are likely to follow a common trajectory if they are all in contact with some other objects and the contact relations are the same. The objects may be influenced in the same way since their interactions are through the contacts with the other objects. We say that such objects form a *spatially correlated objects set* (SCO). Figure 9 shows some examples of SCOs in a typical Angry Birds scenario.

Given a set of initial objects, we obtain the SCOs by checking the node equivalence in the corresponding EGSR network. A node is equivalent to another if the two nodes have the same contact relations with other nodes. Thus the

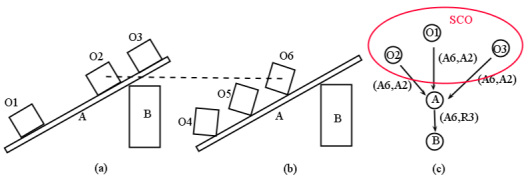

Figure 8: *(a) An initial scene (b) A subsequent scene where the squares have rolled down slightly (c) The EGSR constraint network of the initial scene (only retain the edges indicating contacts) and the SCO*

Figure 9: *(a) A typical Angry Birds scenario (b) SCOs (each highlighted by a different color)*

slope example has only one SCO: $\{o_1, o_2, o_3\}$ (Figure 8.c). Having identified a SCO, we then check the spatial relations between the matched objects in the subsequent scene. Formally, let R be a set of EGSR relations. The converse of a relation $r \in R$ is written as $r' \in R$. Given a SCO in the initial scene $O = \{o_1, o_2, ..., o_k\}$ and a set of subsequent objects $O' = \{o'_1, o'_2, ..., o'_k\}$ with a match, $\forall i \leq k, o_i \sim o'_i$, between them, the spatial constraints can be written as $\forall o_i, o_j \in O, \exists r \in R$ such that $o_i(r)o_j \Rightarrow o'_i(r')o'_j$ *does not hold*, for $i, j \leq k$. If a match violates the constraints, we will try all the other possible matches for the SCO until the violation is resolved. If all matches violates the constraints, we keep the original match. In the slope example, the match $\{o_1 \sim o_4, o_3 \sim o_5, o_2 \sim o_6\}$violates the constraint because o_2(*LEFT*)o_3 and o_6(*RIGHT*)co_5 where *RIGHT* is the converse of *LEFT*.

A Method for Tracking Objects

We proposed a method that utilizes all the above mentioned techniques to solve the matching problem (Alg. 1). We refer to objects in an initial scene as initial objects and objects in a subsequent scene as subsequent objects.

First, the method randomly assigns a unique ID to each initial object, and estimates the MBC quadrants for each initial object according to the object's spatial relations (Alg. 1 line 28). The list of possible matches for each initial object is set so that it contains only the subsequent objects that are of the same type and within the MBC quadrants (Alg. 1 line 3). The method then creates a preference list from the possible matches of each of the initial objects: the subsequent objects in the preference list are sorted by the size of the centroid shift from the initial object in ascending order. The method matches the two sets of objects using a stable marriage algorithm (Gale and Shapley 1962) with the precomputed preference lists (Alg. 1 line 4). The algorithm ensures each match is stable in the sense that no pair of objects would prefer each other over their matched partners. Then, the method finds all groups of spatially correlated objects from the initial objects and gets their corresponding objects from the match (Alg. 1 line 17). The method then checks to see whether the spatial constraint has been violated. If it has,

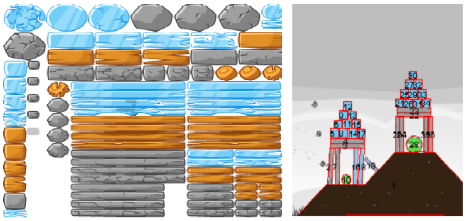

Figure 10: *(a) Main Templates used in Angry Birds (b) The vision detects the real shapes of the objects in a typical Angry Birds scenario*

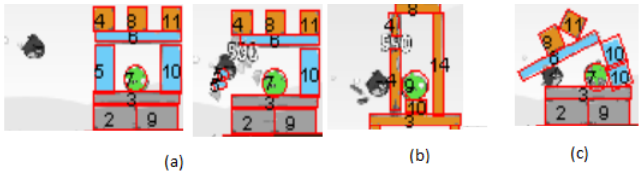

Figure 11: *(a) The object with ID 5 is broken into pieces by a hit (b) The object with ID 4 is partially occluded (c) The object with ID 10 is damaged and detected as two separate blocks*

it resolves this accordingly (Alg. 1 line 34).

Implementation

We implemented our method and applied it to the Angry Birds where the vision system can detect the exact shapes of the objects (Wang 2013). The objects' visual appearance are restricted to a finite number of templates (Figure 10). However, occlusion or fragmentation are not accounted for in the vision system. It has the following limitations that can severely affect the matching accuracy: (1) Debris is not recognized so that it cannot determine whether an object, say a stone, is a real stone or just a piece of debris from a previously destroyed stone (Figure 11.a). (2) Damaged objects will be detected as several separate smaller pieces (Figure 11.b). (3) Objects can be occluded by debris or other game effects e.g. scores (Figure 11.c).

There are several techniques for tackling fragmentation and occlusion (Adam, Rivlin, and Shimshoni 2006; Bose, Wang, and Grimson 2007; Yu, Medioni, and Cohen 2007). Most of them largely depends on their underlying tracking algorithms and their own ad-hoc occlusion reasoning models e.g. inference graph, Bayesian network. We create an approach that can effectively deal with this problem in the Angry Birds domain. Identifying which objects have been destroyed is also important, we show our method can determine this as well in the following section.

Handling Fragmentation and Occlusion

First, we identify all the subsequent objects that could be fragments. We classify the initial and subsequent objects using their templates. For each template T, there is a set T_{ini} of initial objects and a set T_{sub} of subsequent objects with the same template. We treat all objects in T_{sub} as potential fragments if T_{sub} contains more objects than T_{ini}.

Fragments are arranged into groups where all the fragments in a group can form one of the templates. The shape formed by the fragments is an oriented minimum bounding rectangle (OMBR) containing all the fragments (Figure 12.a). We treat the OMBR as one object in the subsequent image, so that it can be matched with an initial object. Once

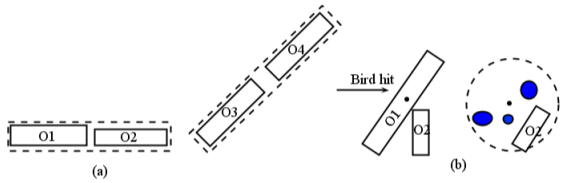

Figure 12: *(a) OMBRs are indicated by the dotted rectangles (b)* o_1 *has been destroyed by a bird hit, and the blue dots are recognized as debris*

the OMBR is matched, the fragments from the corresponding group are also matched and given the same ID. A fragment is not allowed to be in more than one OMBR.

We label the unmatched fragments as debris. Destruction of an object will create debris around the object's location. The debris can be of any shape, e.g. circles or polygons, and will diffuse until it disappear after 1–3 seconds. Given an object o in the initial scenario, we search for its debris if no subsequent objects can be matched with o (including the OMBRs created from the fragments). We first draw the MBC of o. The set of potential fragments are those fragments within the MBC excluding those that have been matched. The set of objects is labelled as debris of o and o is marked as destroyed (Figure 12.b).

An object can also be totally occluded. Before the matching, we cache the spatial configurations of the initial objects. At the end of the matching, we update the cache by replacing each initial object's configuration with that of the matched subsequent object so that the cache always maintains the latest configurations. If an occluded object recurs in a subsequent image, we match the object by searching through the cache for an unmatched initial object. The occluded object will be matched if it lies in the MBC of that initial object. The method determines an object as destroyed if it detects the debris of the object, or when the object has been totally occluded for n second(s). n is tunable and we set n to 1 in the evaluation.

Evaluation

Matching an object can be trivial if the object has a unique appearance or stays stationary across images. We measure the accuracy of the tracking method by the percentage of correct matches out of the possible mismatches. Specifically, given a set n of objects in an initial scenario and assume m of them are either of unique type or stationary across images, then the total number of possible mismatches is n - m. We count the correct matches c of the n - m possible mismatches. The accuracy is $c / (n - m)$. The evaluation has been done in two steps. We first collect samples from active angry birds scenarios, and obtain the ground truth by applying our method using the smallest time gap. Then we evaluate the method by varying the time gaps and obtain the accuracy by comparing against the ground truth.

Obtaining Ground Truth

We collect a sample by capturing a sequence of screenshots before and after a shot using the smallest time gap (50 ms). The average time gap between the beginning and the end screenshot is 10 seconds. That is, a sample contains around 200 screenshots. We apply our method to the whole sequence so that the method will keep tracking the objects through all of the screenshots, from the first until the

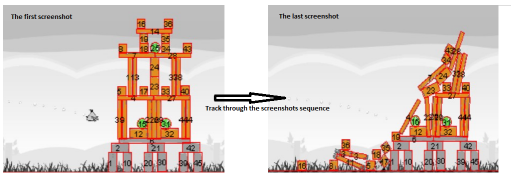

Figure 13: *The method tracks through the screenshots and labels the matched objects with the same ID.*

TPercent	Accuracy		Mismatch	
QSR	QSR	BASIC	QSR	BASIC
0.95	0.91	0.5	2	12
1.00	1.00	1.00	0	0
0.94	0.80	0.40	2	6
0.89	0.85	0.54	2	6
0.91	0.76	0.41	4	10
1.00	1.00	0.00	0	2
1.00	1.00	0.56	0	4
0.93	0.78	0.33	2	6
0.92	0.71	0.42	2	4
0.94	1.00	0.5	0	2

Table 1: *Results on the 10 samples(* QSR*: the proposed method,* BASIC*: the modified method)*

Time gap (ms)	TPercent	Accuracy	Mismatch
100	0.91	0.85	2.22
200	0.89	0.80	3.22
300	0.88	0.77	3.4
500	0.85	0.70	3.98
1000	0.82	0.68	4.22

Table 2: *Results on the 62 samples with different time gaps. TPercen, Accuracy, Mismatch are average of all the samples*

last(Figure 13). The match between the initial objects in the first screenshot and the subsequent objects in the last screenshot will be saved as the ground truth for later evaluation.

To automate this process, we run an angry-birds agent that always aims at an random pig on the first 21 poached-eggs Levels (Rovio 2013). The agent starts to capture screenshots once a shot is made, and stops after 10 seconds. At each level, the agent records the screenshots of at most four shots. We collected 62 non-trivial samples. To evaluate the accuracy of the ground truth, we run our method on 10 samples randomly picked from the 62. We determine the accuracy by manually labelling the initial objects and their correspondence in the end screenshot, and compare with the matching. This accuracy is a lower-bound accuracy of matching between a pair of screenshots with the specified time gap, because any incorrect matches made in the intermediate stage will yield mismatches between the first and end screenshots. The method can match most of the objects with less than 4 mismatches per sample.The method achieves real-time performance with average 3 ms per match. To illustrate the significant improvements achieved by the reasoning techniques mentioned above, we compare our method with an approach (*BASIC*) that matches objects using their visual appearance and minimizes the centroid shift between initial and subsequent objects. *BASIC* is a modified version of our method with the movement approximation, and common movement handling disabled. The results are shown in Table 1 with each row represents a sample. For each sample, we also show the percentage of the number of correctly matched objects out of the total number of objects (*TPercent*).

Experimental Results

We evaluate our method on the 62 samples (12400 screenshots) with varying time gaps, namely 100 ms, 200 ms (the maximum delay in getting screenshots in the competition), 300 ms (the time taken by requesting a screenshot plus the vision segmentation), 500ms, and 1000 ms. For a particular time gap, say 200 ms, the method will start from the first screenshot, go through every *200/50 = 4* (50 ms is the time gap used to collect the samples) screenshots of the original sequence, until the last. E.g. Given a sample with a sequence of screenshots $s_1, s_2, ..., s_{14}$, the method using 200 ms as the time gap will track through s_1, s_6, s_{10}, s_{14} sequentially. The accuracy and mismatches are then obtained by comparing against the saved ground truth. Thus, the mismatches can be viewed as *additional mismatches* to the ground truth. As expected, the accuracy drops down when applying larger time gaps (Table 2).

Conclusion and Future Work

We looked at the problem of tracking multiple objects of the same visual appearance across images. Due to the combinatorial number of possible matches between such objects, this problem gets harder the more time passes between images. We developed a method for solving this problem that is based on a qualitative spatial representation of different object properties. We used the Angry Birds game as an example domain, as object detection in this domain is solved for known objects and the number of possible object types is limited. We showed that our method is very accurate in identifying which object in an "after" image correspond to which object in a "before" image, while shooting birds at the structure that protects the pigs. We tested different time gaps between images and analysed how the time gap affects the accuracy of our method.

Our method should be very useful for the long term goal of building an AI agent that can play Angry Birds better than the best human players. One of the big problems that need to be solved is the prediction of consequences of shots. Our method allows researchers to build sophisticated machine learning algorithms for learning consequences of shots. Instead of just providing a before and after state of shots, we can now automatically identify correspondences between objects in the before and after states, which should lead to much better learning outcomes. As a side-effect it also allows us to identify which objects have been destroyed and which objects haven't. It also allows us to test the success and predicted outcome of Angry Birds game playing strategies, such as the structural analysis developed by Peng and Renz (Zhang and Renz 2014). The time gaps we considered are relevant as they form the minimum time gaps for agents to take and to analyse pictures while actively playing the game. The less often we have to take and analyse screenshots the better. One future goal is to further increase the time gap, but with large time gaps the problem of matching objects is getting extremely hard as there are fewer and fewer cues. We did some initial cognitive studies and asked people to match objects in before and after Angry Birds images. When the time gaps were greater than 2 or 3 seconds, people were mostly unable to find or to explain a correct match.

As mentioned in the introduction, there are a number of other interesting application domains where similar prob-

lems are considered. One problem in other domains might be that object recognition is not as accurate as in the Angry Birds domain, also because the number and type of objects is not restricted. One possible line of future work is to extend our method to deal with arbitrary objects and not only rectangular shaped objects. We might be able to use similar methods by using the tightest fitting minimum bounding rectangles that can have any angle around all objects, independent of the objects' shape and orientation.

Algorithm 1 The Object Tracking Algorithm

```
1: procedure MATCHOBJECTS(objs)
2:     iniobjs// initial objects, sol← {}
3:     pmatches←ApproxMovement(iniobjs,objs)
4:     CalculatePreference(iniobjs,pmatches)
5:     freeobjs←iniobjs
6:     while freeobjs is not empty do
7:         iniobj ← dequeue(freeobjs), get the next preferred obj
       from iniobj's preference list
8:         if obj is not assigned yet then sol ← sol ∪
       {(iniobj,obj)}
9:         else obj has been assigned to iniobj'
10:            if obj prefers iniobj to iniobj' then sol← sol ∪
       {(iniobj, obj)}, freeobjs← freeobjsj'}
11:            else freeobjs← freeobjs∪{iniobj}
12:            end if
13:        end if
14:    end while
15:    CommonMotion(GetSCO(iniobjs), sol)
16: end procedure
17: procedure GETSCO(objs)
18:    cobjsList←{}
19:    for obj∈objs do
20:        cobjs←objs, tobjs←objects that touch obj
21:        for tobj∈tobjs do
22:            ttobjs← a set of the objects excluding obj that
       touch tobj via the same contact.
23:            cobjs←cobjs ∩ ttobjs
24:        end for
25:        cobjsList←cobjsList ∪ cobjs
26:    end for return cobjsList
27: end procedure
28: procedure APPROXMOVEMENT(iniobjs, objs)
29:    for iniobj∈iniobjs do
30:        pobjs←{}, compute the active quadrants of iniobj, add
       obj∈objs to pobjs if obj is within the quadrants and have the
       same type with iniobj
31:        pmatches←pmatches ∪ {(iniobj, pobjs)}
32:    end for return pmatches
33: end procedure
34: procedure COMMONMOTION(cobjsList, sol)
35:    for cobjs∈cobjsList do
36:        for each object in cobjs, get its matched obj from sol.
       Check whether the spatial constraints are violated. Re-assign
       until resolved
37:    end for
38: end procedure
```

References

Adam, A.; Rivlin, E.; and Shimshoni, I. 2006. Robust Fragments-based Tracking using the Integral Histogram. In *Computer Vision and Pattern Recognition*, volume 1, 798–805.

AIBirds. 2013. Angry Birds: AI Competition. http://www.aibirds.org.

Balbiani, P.; Condotta, J.-F.; and Del Cerro, L. F. 1998. A model for reasoning about bidimensional temporal relations. In *PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING-INTERNATIONAL CONFERENCE*-, 124–130. Citeseer.

Belongie, S.; Malik, J.; and Puzicha, J. 2002. Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 24(4):509–522.

Bose, B.; Wang, X.; and Grimson, E. 2007. Multi-class object tracking algorithm that handles fragmentation and grouping. In *Computer Vision and Pattern Recognition*.

Cohn, A., and Renz, J. 2008. Qualitative spatial representation and reasoning. *Foundations of Artificial Intelligence* 3:551–596.

Cohn, A. G.; Renz, J.; and Sridhar, M. 2012. Thinking inside the box: A comprehensive spatial representation for video analysis. In *KR*.

Cutler, R., and Davis, L. S. 2000. Robust real-time periodic motion detection, analysis, and applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22(8):781–796.

Gale, D., and Shapley, L. S. 1962. College admissions and the stability of marriage. *The American Mathematical Monthly* 69(1):9–15.

Ge, X., and Renz, J. 2013. Representation and reasoning about general solid rectangles. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI'13, 905–911. AAAI Press.

Goyal, R., and Egenhofer, M. 1997. The direction-relation matrix: A representation of direction relations for extended spatial objects. *UCGIS annual assembly and summer retreat, Bar Harbor, ME* 65–74.

Lowe, D. G. 1999. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, 1150–1157. Ieee.

Papageorgiou, C. P.; Oren, M.; and Poggio, T. 1998. A general framework for object detection. In *Computer Vision, 1998. Sixth International Conference on*, 555–562. IEEE.

Pollock, J. L. 1974. *Knowledge and justification*. Princeton University Press Princeton.

Rovio. 2013. Angry Birds Chrome. http://chrome.angrybirds.com/.

Santore, J. F., and Shapiro, S. C. 2005. *Identifying perceptually indistinguishable objects*. State University of New York at Buffalo.

Sokeh, H. S.; Gould, S.; and Renz, J. 2013. Efficient extraction and representation of spatial information from video data. *Proceedings of IJCAI-13*.

Sonka, M.; Hlavac, V.; Boyle, R.; et al. 1999. Image processing, analysis, and machine vision.

Sridhar, M.; Cohn, A. G.; and Hogg, D. C. 2011. From video to rcc8: exploiting a distance based semantics to stabilise the interpretation of mereotopological relations. In *Spatial Information Theory*. Springer. 110–125.

Viola, P.; Jones, M. J.; and Snow, D. 2005. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision* 63(2):153–161.

Wallgrün, J. O.; Wolter, D.; and Richter, K.-F. 2010. Qualitative matching of spatial information. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 300–309. ACM.

Wang, A. 2013. Angry birds project (vision component). http://aibirds.org/andrew_wang_report.pdf.

Yilmaz, A.; Javed, O.; and Shah, M. 2006. Object tracking: A survey. *Acm Computing Surveys (CSUR)* 38(4):13.

Yilmaz, A.; Li, X.; and Shah, M. 2004. Contour-based object tracking with occlusion handling in video acquired using mobile cameras. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 26(11):1531–1536.

Yu, Q.; Medioni, G. G.; and Cohen, I. 2007. Multiple Target Tracking Using Spatio-Temporal Markov Chain Monte Carlo Data Association. In *Computer Vision and Pattern Recognition*.

Zhang, P., and Renz, J. 2014. Qualitative spatial representation and reasoning in angry birds: First results. In *KR*.