Multiple Object Tracking using Spatial Reasoning

Xiaoyu Ge and **Jochen Renz**
Research School of Computer Science
The Australian National University
Canberra, Australia
{xiaoyu.ge, jochen.renz}@anu.edu.au

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.

In this paper, we propose a solution to the problem of tracking multiple objects across images taken at different time points. We represent objects using general solid rectangles, i.e., rectangles that can have any angle and are not penetrable (GSRs). GSRs are often used in computer vision and video games. We present a modified version of the GR-n representation [Ge and Renz, IJCAI 2013] that allows us to specify useful spatial constraints and infer possible spatial changes of GSRs.

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. We also show that our solution 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 eyes, cameras, or other optical devices to "see" their environment and to detect changes in their environment. The information these devices provide is in the form of image sequences, or if the image sampling rate is high enough, we call it video. Image understanding (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), that is the ability to identify the same object in a series of images or in video and to track its movement and its changes. Existing object tracking methods typically rely on the visual appearance of objects and on their trajectories to successfully track even temporarily occluded objects (Yilmaz, Li, and Shah 2004; Cutler and Davis 2000; Viola, Jones, and Snow 2005). Similar methods are used for multiple object tracking (Berclaz et al. 2011; Yang, Duraiswami, and Davis 2005; Han et al. 2004). In this paper we are interested in a problem that can occur when intelligent agents observe the world through visual perception and have to track changes. It can be described as follows: Given two images, let's call them "before" and after", that depict the same scene at different times. Between the two images, certain physical actions have been happening that affected the locations and possibly the states of the objects in the scene. Our task is to find a match between the objects in the "before" image with the objects in the "after" image that is consistent with the effects of the physical actions that happened between the two images. In case there are multiple consistent matches, we want to identify the most plausible one.

It is clear that this problem is relatively straightforward to solve if the visual appearance of all objects in each image is unique, as we simply match the objects that have the same visual appearance. The problem gets challenging, if multiple objects have the same visual appearance. We say that objects of the same visual appearance have the same type and we can only match objects of the same type. Whenever there are multiple objects of the same type, existing object tracking methods are not applicable any more.

There are different variants of this problem depending on the way the relevant physical actions are specified or unspecified. In general, the actions pose additional constraints that have to be satisfied. These constraints can vary from case to case and typically depend on the application. One variant of the problem is when the specific actions that are responsible for the changes are unknown. Then the task is to find a match for which (THIS IS REALLY WEIRD THAT WE DO NOT TAKE INTO ACCOUNT THE PARTICULAR ACTION!!!) the standard laws of physics are satisfied. ARE WE REALLY DOING THIS, DOES THIS MAKE SENSE? We could also formulate the problem as identifying physical actions that explain a given match. Depending on the speed of the changes, it is clear that the longer the time gap between the "before" and the "after" image, the less accurate a match might be.

There are many real-world situations where this problem occurs. For example in natural disasters such as earthquakes, storms, or tsunamis, where we often see before and after images, also in terrorist attacks or bomb explosions. But there are also less dramatic situations, for example a satellite that takes another image of the same site at the next fly-over.

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 Angry Birds as good as the best human players. One major problem in this context is to be able to accurately predict the outcome of an action. To do this, we need to learn from previous cases where the effect an action had on a given scenario is known. As input to the machine learning algorithms we need to know the initial scenario, the action that was used and the effect the action had on each object in the initial scenario, i.e., we need to know exactly which object in the initial state corresponds to which object in the resulting state. In order to be able to learn a huge number of cases, we need to be able to automate the matching between objects in the "before" and in the "after" image.

We evaluate our proposed solution using the Angry Birds scenario. We take two subsequent screenshots of an active Angry Birds game with varying time gaps and apply our method to match the objects between the two 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.

Preliminary

- Qualitative Spatial Reasoning

"Our method is based on qualitative spatial calculi and reasoning techniques developed in the qualitative spatial reasoning (QSR) community."

We focus on rectangles and circles only.

Objects Representation with the extended GSR-n (EGSR)

We use a minimum bounding rectangle (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 impenetrable.

Many rectangle-based spatial calculi (Balbiani, Condotta, and Del Cerro 1998; Cohn, Renz, and Sridhar 2012; Sokeh, Gould, and Renz 2013) are developed in the context of Qualitative Spatial Reasoning for dealing with different spatial aspects of such as topology, size etc. One of them is GSR-n, 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. Given two GSRs o_1 and o_2 that contact each other via $sector_1, sector_2 \in \{S_1, ..., S_8, R_1, ..., R_8\}$, 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). 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 by the $Qualitative\,Corner\,Instantiations$, which qualitatively describes the size and leaning direction of a GSR. Since we represent the rectangles using real shapes, we do not need the unary relations.

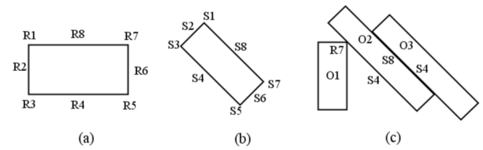

Figure 1: Contact sectors of (a) a normal rectangle (without rotation) and (b) an angular rectangle. (c) An example scenario where $o_1(R_7, S_4)o_2$, $o_2(S_8, S_4)o_3$

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 which is 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 . 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 this new representation EGSR (see Figure 2). All EGSR relations are obviously converse, e.g. the converse of $TOP\,LEFT$ is $BOTTOM\,RIGHT$, the converse of (R_4, S_1) is (S_1, R_4).

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.

(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 3 shows an example of a QCN based on EGSR relations.

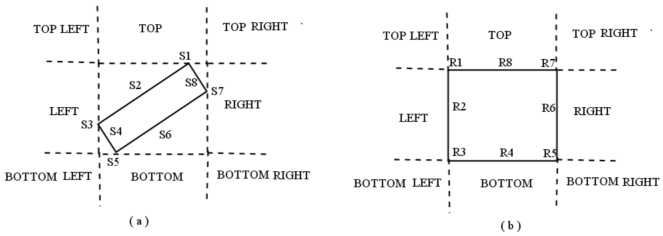

Figure 2: Contact sectors and Cardinal directions of (a) an angular rectangle and (b) a normal rectangle

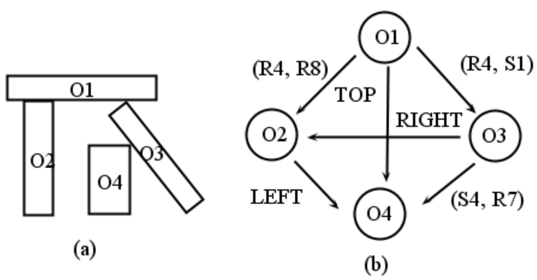

Figure 3: (a) A spatial scenario where the four rectangles form a stable structure under downward gravity (b) The corresponding qualitative constraint network

Efficient Matching by Approximating Movement

To find out the best matchings, a straightforward approach is to enumerate all the possibilities. The search space 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 MBC can be divided into four quadrants to further restrict the search area. A quadrant is said to be active if all the objects in that quadrant can be considered as potential matches of the referred object, otherwise the quadrant is inactive. An initial object will be matched with one of the subsequent objects within the active quadrants while the others will not be considered.

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 .a). $(*,*)$ refers to an arbitrary quadrant.

The relative distance between two objects can be allocated to three meaningful classes, namely touch, reachable, and non-reachable (Figure .b). 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.

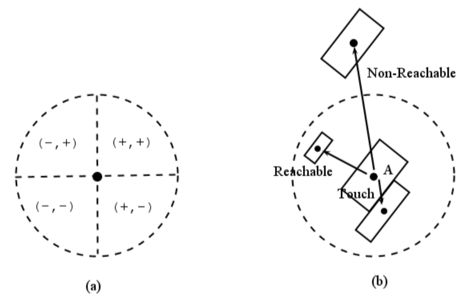

Figure 4: (a) The four quadrants of a MBC (b) Qualitative distance with respect to object A and its MBC

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. An unstable object can have one of two possible motions, free fall if it is unsupported or fall to the side where there are no supports.

We call the movement approximating problem we want to solve MQSR(Θ, QCN, M) where Θ is a set of objects in a spatial scenario and QCN is the corresponding EGSR constraint network over the objects. M is a reasoning model composed of a set of configurations written in EGSR relations 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 and QCN against M.

We demonstrate the movement approximation in the Angry Birds scenario where a bird hit usually comes from the left. The model we use is mainly based on stability analysis.

(Ge and Renz 2013) defined four kinds of supports that can make a solid rectangle stable and provided the corresponding GSR configurations. A simple model M can contain all such stable configurations. Given each object, the model will evaluate the stability of the object by the configurations and return $C^{(*,-)}$ for those unstable ones.

A stable object may become unstable if it loses a support due to a bird hit, and this may create a chain of effects if the object also supports other objects. 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 5).

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 $C^{(+,-)}$ / $C^{(-,+)}$ (Figure 6). 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. Then the following two configurations that can be added to the model M:

1. $\forall o_1^R : \exists o_2^* \, o_1^R(S_5,*)o_2^* \wedge \neg\exists o_3^* \, o_1^R(S_6,*)o_3^* \wedge \neg\exists o_4^* \, o_1^R(S_7,*)o_4^* \Rightarrow C_{o_1^R}^{(+,-)}$
2. $\forall o_1^L : \exists o_2^* \, o_1^L(S_5,*)o_2^* \wedge \neg\exists o_3^* \, o_1^L(S_3,*)o_3^* \wedge \neg\exists o_4^* \, o_1^L(S_4,*)o_4^* \Rightarrow C_{o_1^L}^{(+,-)}$

We will show the model M, although simple, is sufficient to solve the tracking problem in the evaluation part.

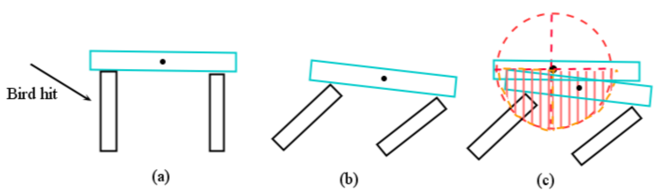

Figure 5: (a) The object in cyan is supported via one-edge support, and the impending impact is indicated by the arrow (b) A subsequent scene after the bird hit (c) The estimated active quadrant (shadowed area)

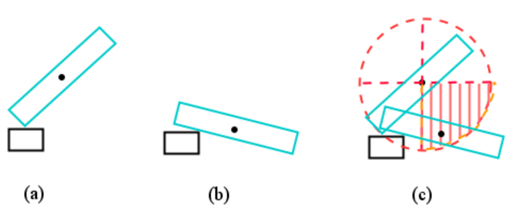

Figure 6: (a) The right-leaning object (in cyan) (b) a subsequent scene where the object falls to the right (c) The estimated active quadrant (shadowed area)

Handling Common Movement by Spatial Reasoning

One challenge is to determine a match between identical objects 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 identical 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. 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 inverted 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 inverted 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 object* (SCO). Figure 7 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 slope example has only one SCO: $\{o_1, o_2, o_3\}$ (see Figure 8.c).

Having identified an SCO, we then check the spatial relations between the matched objects in the subsequent image. 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 group objects that form a spatially correlated object in the initial image $O = \{o_1, o_2, ..., o_k\}$ 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.

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 $o_6\{RIGHT\}o_5$ where $RIGHT$ is the converse of $LEFT$.

A Method for Objects Tracking

Let O and O' be a set of objects in an initial image and a subsequent image taken at a later time, respectively. For ease of notation, we will refer to objects in O as initial objects and objects in O' as new objects.

Figure 7: (a) A typical Angry Birds scenario (b) SCOs (each highlighted by a different color)

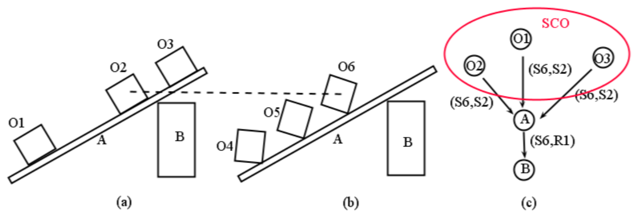

Figure 8: (a) An initial scene (b) A subsequent scene where the squares have rolled down slightly (c) The EGSR constraint network of the initial scene (only retain the edges indicating contacts) and the SCO

First, the method randomly assigns a unique ID to each initial object, and estimates the MBC quadrants of each object according to the object's spatial relations. The domain for each initial object is set so that it contains only the new objects that are of the same object type and within the MBC quadrants. The method then creates a preference list from the domain of each of the initial objects: the new 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 (**?**) with the pre-computed preference lists. 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 among the initial objects and gets their corresponding objects from the match. The method then checks to see whether a spatial constraint has been violated, as mentioned in . If there has, it resolves it accordingly.

Implementation

We implemented our method and applied it to the Angry Birds where the vision system can detect the exact shapes of the objects (**?**). The objects' visual appearance are restricted to a finite number of templates (see Figure 9).

The vision has the following limitations that can severely affect the matching accuracy:

- Misdetection of damaged objects. Damaged objects will be detected as a number of separate smaller pieces. (Figure 10.b)
- Debris is not recognized so that the system cannot determine whether an object, say a stone, is a real stone or just a piece of debris from a previously destroyed stone (Figure 10.a).
- Occlusion of objects is not handled. Objects can be partially or entirely occluded by debris or other game effects e.g. scores or clouds around the hit point (Figure 10.c)

The fragmentation of objects creates new objects in subsequent images and the new objects are not directly related to the initial ones. The varying number of objects from one scenario to another can introduce ambiguities which can confuse the tracking algorithm.

There are several approaches for tackling this problem.(Adam, Rivlin, and Shimshoni 2006; Bose, Wang, and Grimson 2007; Yu, Medioni, and Cohen 2007). Most of them largely depends on their underlying tacking algorithms and their own ad-hoc occlusion reasoning models e.g. inference graph, Bayesian network.

We create an approach that can effectively deal with the fragmentation in the Angry Birds context where objects are created on a finite number of templates.

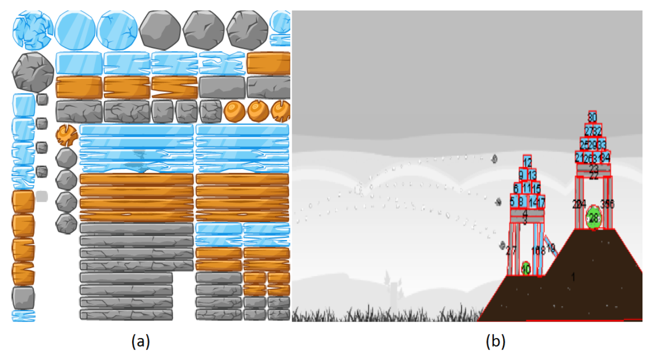

Figure 9: (a) Main Templates used in Angry Birds (b) The vision detects the real shapes of the objects in a typical Angry Birds scenario

Handling Fragmentation and Occlusion

In Angry Birds, fragmentation is mainly due to the destruction of objects, partially occlusion, and damage. By recognizing these fragments, the system is able to infer whether an object has been destroyed, damaged, or occluded, which leads to robust tracking. We achieve this in three steps:

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

Fragments are then arranged in 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) that bounds all the fragments (see Figure 11.a). We then treat the OMBR as one object in the subsequent image, so that it can be matched with one of the initial objects. Once the OMBR is matched, all the fragments from the corresponding group are also matched, i.e. assigned with

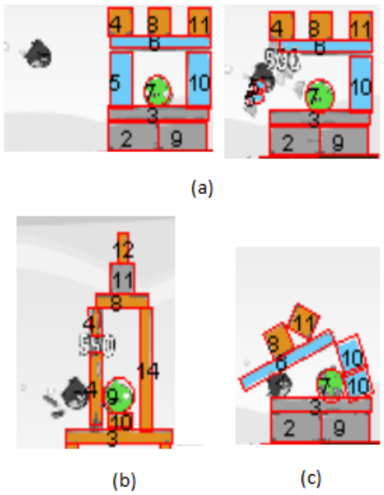

Figure 10: (a) The object with ID 5 is broken into pieces by a hit (b) The object with ID 4 is partially occluded (c) The object with ID 10 is damaged and detected as two separate blocks

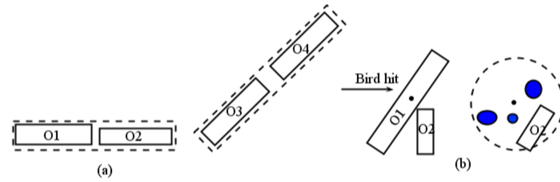

Figure 11: (a) OMBRs are indicated by the dotted rectangles (b) o_1 has been destroyed by a bird hit, and the blue dots are recognized as debris

the same ID. In some rare cases, a fragment may be included in more than one OMBR. So we add an additional constraint that a fragment is not allowed to be common to more than one OMBR.

We label the unmatched fragments as debris. Destruction of an object will create a cluster of debris around the object's location. The pieces of debris can be of any shape, e.g. a circle or polygon and will diffuse until they 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 the objects is labelled as debris of o and o is marked as destroyed (see Figure 11.b). In this process, some pieces of debris might be from more than one initial objects and we do not explicitly label pieces of debris with an ID.

An object can also be totally occluded. To deal with this, before the matching, we cache the spatial configurations of all the initial objects. At the end of the matching, we update the cache by replacing each initial object's configuration with the matched subsequent object's so that the cache always maintains the latest configuration of each initial object. 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.

Evaluation

Matching an object can be trivial if the object has a unique appearance or the object stays stationary across the images. We measure the accuracy of the tracking method by the percentage of correct matches out of the non-trivial 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 among the $n-m$ objects which can be potentially mismatched. The accuracy is $c/(n-m)$.

The evaluation has been done in two steps. In the first step, we collect samples from active angry birds scenarios, and obtain the ground truth by applying our method using the smallest time gap. In the second step, 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 last(see Figure 12). 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.

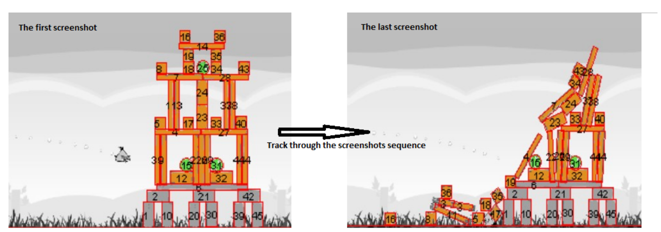

Figure 12: The method tracks through the screenshots and label the matched objects with the same ID. The match between the first and last screenshots is saved as the ground truth

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 removed all the samples where all the objects are stationary (the bird did not hit anything). Finally we have 62 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.

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 spatial reasoning, we compare our method with an approach ($BASIC$) which matches objects by visual appearance and minimizes the centroid shift between initial and subsequent objects. The results are shown in Table 1.

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

	Accuracy		Mismatch	
Sample	QSR	BASIC	QSR	BASIC
1	0.91	0.5	2	12
2	1.00	1.00	0	0
3	0.80	0.40	2	6
4	0.85	0.54	2	6
5	0.76	0.41	4	10
6	1.00	0.00	0	2
7	1.00	0.56	0	4
8	0.78	0.33	2	6
9	0.71	0.42	2	4
10	1	0.5	0	2

Experimental Results

We evaluate our method on the 62 samples (12400 frames) with varying time gaps, namely 100 ms, 200 ms (the maximum delay in getting screenshot from the server used in Angry Birds AI competition), 300 ms (the time taken by requesting a screenshot plus the vision segmentation) and 1000 ms

For a particular time gap of size T, the method will track through every $T/50$ screenshots of the original sequence. The accuracy and mismatch is obtained by comparing against the saved ground truth. As expected, the accuracy drops down when applying larger time gaps (see Table 2).

Table 2: Results on the 80 samples with different time gaps

Time gap (ms)	Average Accuracy	Average Mismatch
100	0.85	2.22
200	0.80	3.22
300	0.77	3.4
500	0.70	3.98
1000	0.68	4.22

Conclusion and Future Work

Algorithm 1 The Object Tracking Algorithm

procedure MATCHOBJECTS($objs$)
    $iniobjs$ //initial objects
    $objs \leftarrow$ HandlingFragments($objs$) // add OMBRs created from potential fragments
    $solution \leftarrow \{iniobjs, \{\}\}$
    $domain \leftarrow \{iniobjs, \{\}\}$// A map stores each initial object as key, and its list of possible corresponding objects
    $domain \leftarrow$ MotionApproximation($iniobjs$, $objs$)
    CalculatePreference($iniobjs$, $domain$) //Calculate the preference list of $iniobjs$
    $freeobjs \leftarrow iniobjs$
    while $freeobjs$ is not empty **do**
        $iniobj \leftarrow dequeue(freeobjs)$
        get a next preferred obj from $iniobj$'s preference list
        if obj is not assigned yet **then**
            match($iniobj$, obj)
        else obj has been assigned to another initial object $iniobj'$
            if obj prefers $iniobj$ to $iniobj'$ **then**
                match($iniobj$, obj), $freeobjs \leftarrow freeobjs \cup \{iniobj'\}$
            else
                $freeobjs \leftarrow freeobjs \cup \{iniobj\}$
            end if
        end if
    end while
    $cobjsList \leftarrow$ GetSCO($iniobjs$)// get spatially correlated objects
    CommonMotion($cobjsList$,$solution$), find the debris, $iniobjs \leftarrow objs$
end procedure
procedure GETSCO($objs$)
    $cobjsList \leftarrow \{\}$, $QCN \leftarrow$ construct a QCN on $objs$
    for $obj \in objs$ **do**
        $cobjs \leftarrow objs$
        $tobjs \leftarrow$ a set of the objects that touch obj
        for $tobj \in tobjs$ **do**
            $ttobjs \leftarrow$ a set of the objects excluding obj that touch $tobj$ via the same contact relation.
            $cobjs \leftarrow cobjs \cap ttobjs$
        end for
        $cobjsList \leftarrow cobjsList \cup cobjs$
    end forreturn $cobjsList$
end procedure
procedure MOTIONAPPROXIMATION($iniobjs$, $objs$)
    for $iniobj \in iniobjs$ **do**
        $domain \leftarrow domain \cup \{iniobj, \{\}\}$, $pobjs \leftarrow \{\}$, compute the active quadrants of $iniobj$
        for each $obj \in objs$, add obj to $pobjs$ if obj is within the quadrants and have the same type with $iniobj$
        $domain \leftarrow domain \cup \{obj, \{pobjs\}\}$
    end forreturn $domain$
end procedure
procedure MATCH($iniobj$, obj)
    $obj.id \leftarrow iniobj.id$, $solution \leftarrow \{iniobj, obj\}$
end procedure
procedure COMMONMOTION($cobjsList$, $solution$)
    for $cobjs \in cobjsList$ **do**
        for each object in $cobjs$, get its matched obj from $solution$. Check whether the spatial constraints are violated. Re-assign until resolved
    end for
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.

Berclaz, J.; Fleuret, F.; Turetken, E.; and Fua, P. 2011. Multiple object tracking using k-shortest paths optimization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 33(9):1806–1819.

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. 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.

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.

Han, M.; Xu, W.; Tao, H.; and Gong, Y. 2004. An algorithm for multiple object trajectory tracking. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, I–864. 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.

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

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

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. 110125.

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.

Yang, C.; Duraiswami, R.; and Davis, L. 2005. Fast multiple object tracking via a hierarchical particle filter. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, 212–219. IEEE.

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*.