AI for Herding Sheep

Peter Cowling and Christian Gmeinwieser

AI Research Centre
School of Computing Informatics and Media
University of Bradford
Bradford BD7 1DP
p.i.cowling@bradford.ac.uk, c.gmeinwieser@gmail.com

Abstract

Shepherding with a dog presents an interesting challenge for artificial intelligence, with multiple intelligent systems assessing and interacting with each other in order to achieve a variety of goals. We present a solution to this problem, which consists of a dog AI making use of influence mapping, state machines and A* pathfinding to respond intelligently to real-life shepherding commands issued by a high-level shepherd AI steering the flock of sheep through waypoints on a variety of maps by using pathfinding and influence maps. The role of the AI shepherd can also be taken by a human player (using either a point and click or voice recognition interface) for matches against the artificial shepherd which proved to be a worthy opponent for human testers. The system was evaluated through user testing and provided a high degree of realism and engaging gameplay relying heavily on the workings of the presented AI components.

Introduction

Shepherding, as far as this paper is concerned, involves controlling a flock of artificial sheep with an artificial sheepdog, by making use of the sheep's predator avoidance behaviors. In the simulation presented here, the dog responds to standard real-life shepherding commands, issued by either an artificial or human shepherd.

While accurate simulation of shepherding behaviors is immediately applicable to games having this as the main gameplay idea such as the small application developed for testing the concepts described here, it also has the potential to lead to the integration of interesting gameplay mechanics in a wide range of other genres such as strategy or first person shooters, where a player indirectly controls a fairly intelligent AI agent (another prominent example here is Black and White [Molyneux, 2001] in which players are assisted by a trainable creature which can interact with the worlds human population)

In the game demo human players can take the role of the shepherd with the goal of guiding sheep through a course as quickly as possible. This role can also be taken by an AI player, which can either be watched alone or challenged in split screen mode. Keyboard and mouse input on the PC is accompanied by voice recognition capabilities to provide a more natural way of issuing commands.

Related work

Moving NPCs to given areas by making use of their steering behaviors is a frequently reoccurring gameplay mechanic since the early days of video games. In Robot Chase (Phipps Associates, 1982), robots chasing the player had to be tricked into electric fences. More recent examples of related gameplay mechanics are Herdy Gerdy (Core Design, 2002), Dog's Life (Frontier Development, 2003) and the goat herding minigame in The Legend of Zelda Twilight Princess (Nintendo, 2006) in which goats need to be assembled in a goal area. Flock! (Capcom, 2009) combined puzzle elements with steering groups of different animals through levels. In these games, players always control the object corresponding to the sheepdog directly.

(Vaughan et al., 1998) conducted work on a robot controlling a flock of live ducks after testing the behavior in simulation, with the robot successfully steering the flock to a goal position in an unobstructed test environment through simple steering behaviors. (Burke et al., 2001), presented a sheepdog installation in which users could interact with a voice controlled dog as a possible application for their c4 architecture for virtual creatures. The main achievement of c4 was to detail a flexible framework for simulated creatures, focusing on realistic perception through pattern recognition and the ability to train the virtual animals which selected from actions based on a reward system. Shepherding specific problems such as splitting sheep into groups or finding an optimal route for the dog to steer the sheep in complex environments have been discussed in detail in (Lien et al., 2004, 2005). Here,

milestones were calculated for the path leading the flock to the goal area as quickly as possible. The artificial shepherd moved to a position behind both the flock and the next milestone to steer the flock into the direction of the next waypoint. In order to minimize sheep disturbance, Lien et al. preferred paths which required the steering AI to stay outside a safe zone, based on the convex hull around all flock members or an enclosing circle, by dynamically constructing waypoints biased towards the edge of the flock. During steering, the shepherd pushed the flock towards the next waypoint with alternating side to side movements, so as to prevent the flock from spreading out, and ultimately fragmenting. At corners, initiating fluid turns was found to lead to more efficient cornering behavior than stopping the flock at the waypoint before steering it into the new direction.

Different flock groups were calculated by first splitting the entire herd into subgroups based on visibility queries from one flock-member to the others. This alone however proved insufficient as the densities of groups can still be very low, making it hard to keep the group together. (Lien et al., 2005) use what they call a compact area which is the smallest possible circle around a group's centroid which could contain all the individuals of that group.

The work presented here builds on this research and splits the shepherd as described by Lien et al. into two distinct entities, a dog which responds to a number of commands commonly used for shepherding and the actual shepherd who issues these commands, controlled by either an AI or a human player. Also, we focus on how the desired behavior can be implemented in an actual game environment through the combination of proven AI techniques including influence mapping. In order to more accurately model the animals involved in the simulation we gathered information related to their behavior. (Shulaw, 2005) describes sheep as animals with a strong flocking instinct, which have evolved a very wide vision cone through which they can perceive their environment and keep track of predators. Binocular vision and therefore depth perception however is limited to less than 50 degrees, which is why sheep face potential predators if not threatened directly. Fleeing is initiated if the position of the intruder is inside the circular flightzone of a sheep, which makes effective herding possible, as not directly affected animals will join the agitated animal in its escape (Grandin 1989).

Through a variety of voiced or whistled commands, a range of different behaviors can be triggered in trained herding dogs, allowing shepherds to gather and move sheep from one place to the other. Commands like "Come-by" and "Away", move the dog around currently focused flock of animals in a clockwise or counter-clockwise direction. "Get out" and "Walk up" are used to increase and decrease the distance of the dog to the flock. "Take time" and "Stand" control the speed of the dog making it either slower or stop completely. In order to make the dog collect more sheep into one place, a command such as "Look back" or "Cast" is usually used. (Raine, 2001)

Game environment

A game world is two dimensional with terrain split into passable land and impassable water and contains one dog, multiple sheep as well as one or several checkpoints and walls (see figure 1). For pathfinding and a large part of the decision making of the AIs, the world is divided into a number of equal-sized squares and information about the current contents and traversability of each square can be quickly queried (see figure 2).

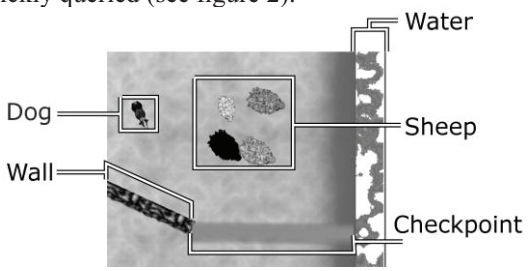

Figure 1: Graphical representation of game elements

The goal for human and AI players is to steer the sheep through checkpoints in sequence on each level. Two types of checkpoints were used, one "race type" defined by two points through which all sheep need to pass through and one "gathering type" defined by a circle in which all of the sheep need to be assembled at once in order to progress. The game is controlled solely by issuing real-life sheepdog commands. Issuable commands are "Away" and "Come-By", "Walk up", "Take time", "Look back" "Get out" and "Stand". Making the dog respond appropriately to these commands needs a substantial component of AI.

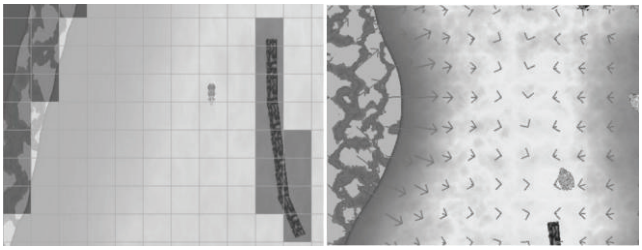

Figure 2: Map Information with grid segmentation and impassable squares on the left and the vector field for height information on the right

As suggested in (Yiskis, 2004; Laming, 2008), the update loop for all actors is split into four different methods which are called in sequence, and which are implemented differently for each actor type. The first updates the character's perception of the world and also performs any necessary planning, for example setting

currently targeted sheep for the dog or placing orders for paths. Then, a "Think" method uses steering behaviors as described in (Reynolds, 1987) to determine the acceleration that should be applied to the actor in order for it to, for example, follow a path or display flocking behavior. The movement layer applies this acceleration to the character's speed, taking into account any character inherent constraints such as mass and speed limits. Finally, collisions with other actors and the environment are resolved to determine the final position of the actor after the current update. To allow for better performance and scalability, not all update steps are performed for each actor every update.

Our sheep are modeled to follow the ethology of their real counterparts. Therefore, sheep follow basic flocking rules such as separation alignment and coherence (Reynolds, 1987). However, the weights for these and other steering behaviors such as fleeing, wall avoidance, keeping the dog in sight, following an assigned mother sheep and moving to higher places by following a vector flow field (Alexander, 2006) with the height information of the level are set by a single layer state machine which changes states depending on a stress level. Therefore sheep can be either grazing, staring at a close dog, fleeing from it if it has invaded the flight zone and therefore raised the stress level, or panicking after having been consistently pushed into a wall or water and caring about nothing else but reaching a safe place The point to which to flee is calculated by examining all squares from a limited breadth first search from the sheep's current position. Physical attributes of sheep such as size and mass affect acceleration and obstacle avoidance and are set randomly at level start within realistic limits.

Sheepdog AI

To achieve adequate command response in all situations, the dog uses a hierarchical and stack-based finite state machine (Fu & Houlette, 2004) to receive and act on commands by setting action states and calculating appropriate paths. At the top layer, the dog is in one of seven different behavior states, which are responsible for handling commands and if necessary set and update the dog's path. Additional states such as a "get within range" state can be pushed if the dog cannot perform a command at its current distance to sheep (see figure 3).

The dog makes use of three important groups of sheep, the closest group, the main sheep group to which fetched sheep are returned and the currently steered sheep group. These groups and the steering target can be set, updated and accessed through the different behavior states as needed in order to fulfill the current command. Clustering of sheep into groups is performed by starting a group with the first sheep and adding any of its perceived neighbors which are within a predefined minimal distance to the group, continuing the process until none of the yet unassigned sheep are within range and starting a new cluster with the first unassigned sheep if necessary.

An influence map based on a square's proximity to the closest sheep was used for finding paths preventing the dog from coming too close to sheep. Influence maps provide a simple and effective way to provide AI with useful information about the environment.(Miles & Louis, 2006) To calculate sheep influences, we add all squares currently containing sheep as starting points for a breadth first search, limited in depth by the maximal distance to which influences are considered. If, during this search, a square is entered from two different ways, it takes on the lower G-value. After the search has investigated every square within this limit, the G-value of all squares on the closed list is subtracted from the inter-square distance in an A* algorithm, resulting in the new maximal influence value for sheep. All paths followed by the dog are reduced using "String pulling" or "line-of-sight testing" (Tozour, 2004), which was modified for paths for which it was relevant to take into account sheep proximity so that only nodes are removed that are not necessary to maintain the low disturbance value of the original path.

Due to the dynamic nature of the game and paths usually being updated only every few seconds, further dynamic obstacle avoidance was applied to keep the dog from running into sheep in tight situations.

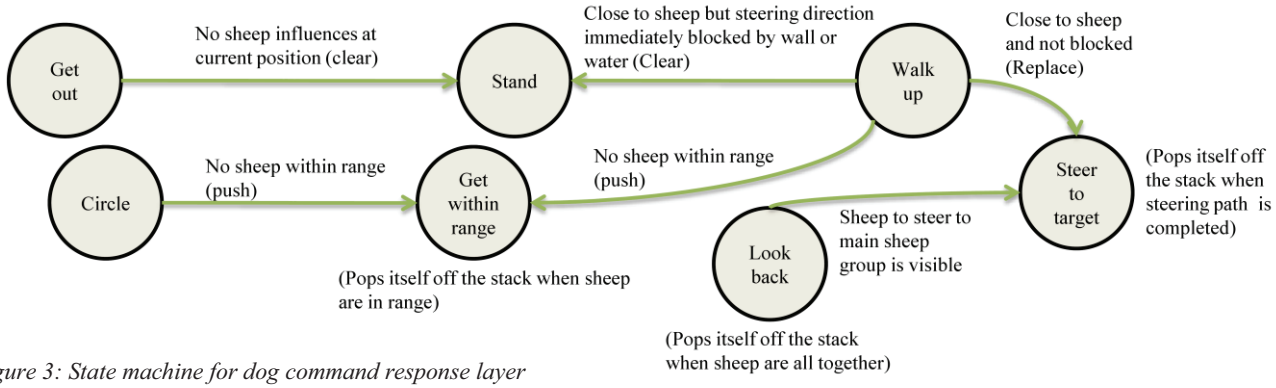

Figure 3: State machine for dog command response layer

Circling

Circling paths need to guide the dog around the sheep in the direction of the command and are updated frequently to adjust to sheep movement. The goal position is calculated by finding a point lying on a line away from the flock's centroid perpendicular to the vector between the dog and the group's centroid. The distance of this point to the centroid is calculated by trying to achieve the same distance the dog has at its start position to the nearest sheep there to the nearest sheep at the goal position, limited by the presence of walls or water.

Clockwise or counter clockwise movement for circling a group of sheep was enforced by increasing the neighbor distance of squares where the motion from the parent is opposed to the desired direction so that paths circling the flock in the right direction are valued as "shorter" and therefore found first if they exist. To implement this, we determine the dot product of the directions to the center of the flock from P and its parent rotated by 90°. With a result smaller than zero, the motion is clockwise while a positive result corresponds to anti-clockwise motion. To limit flooding for circling paths, which can be substantial due to the larger G values from the use of the influence map (a problem also described in Paanacker, 2008), we used a third modifier that penalizes squares without any sheep influence, which was possible as the distance to keep from sheep is smaller than the distance to which sheep influences are applied. The final circling related modifier applies a penalty if the path node lies within a predefined radius from the centre point of the search, resulting in reasonable paths even if the dog was close to the center of the sheep group when paths were calculated.

Casting and Steering

Upon entering the look back state, the dog attempts to find a path to a sheep outside the main sheep group, making the group containing this sheep the currently targeted sheep group of the dog's perception object. During updates the visibility between the dog and the stray sheep is checked, and if the stray sheep is visible, the steering target of the dog's perception is set to the centroid of the main sheep group. After this, the steer to target state is pushed on top of the stack, which takes care of steering the targeted sheep group to the goal. The targeted sheep group and steering target properties of the dog's perception object are used to calculate a steering path from the centroid of the targeted sheep group to the steering target. After this path has been found, the dog follows a path to a steering position from which the sheep are moved closer to the next waypoint (Figure 4).

This steering position is calculated by first finding the sheep of the currently targeted group farthest in Euclidean distance from the next waypoint of the steering path and calculating a position slightly inside that sheep's flight zone in order to steer it closer. If a steered sheep does not move exactly towards the next waypoint, the steering position was optimized to take into account inertia by offsetting it so that it would steer the sheep into the direction given by the heading mirrored across the direction to the waypoint if there would be no inertia.

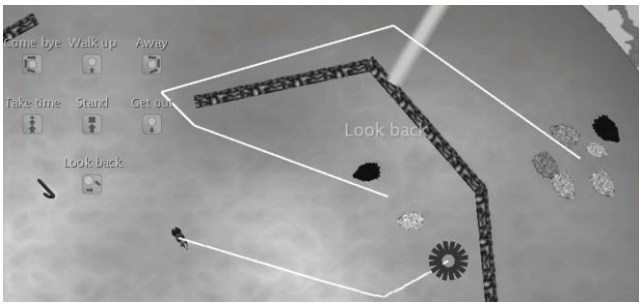

Figure 4: Casting example with highlighted steering and approach path

Targeting the farthest sheep and counteracting for inertia helped to keep the flock together and in many situations automatically results in the side to side described in (Lien et al., 2004). When the centroid of the steered sheep is close to the final node of the steering path, this state is popped from the stack.

The steer to target state is alternatively entered through the "walk up" state responding to a "walk up" command. A dog entering this state will find a path towards the closest sheep. When inside this sheep's flight zone, the steering target is set to the end of a Bresenham line from the dog's position in the direction of the centroid of the closest sheep group until it is blocked by obstacles.

Other Command Responses

The dog responds to a "Get out" command by placing a path order for a path guiding it away from sheep whilst avoiding disturbance. Once the end of this path has been reached and there are no sheep around, the dog resumes with the previous command. Otherwise, the path is updated periodically so that the options for a better goal position of the path in the current situation can are reconsidered. The "Take Time" and "Stand" commands slow down the speed of the dog or stop it completely.

Shepherd AI

The artificial shepherd controls the game by placing commands to the dog in the same way human players do. This was preferred to a fully automated dog, (as has been achieved in other studies with related goals (Vaughan et al. 1998; Lien et al. 2004) as this keeps the shepherd and dog separate, which is more realistic for the simulated environment with herding dogs and allows for new gameplay with human players who can compete against the AI shepherd on equal terms. Commands are placed using a

timer to simulate the reaction time of a human player. This provides a natural-looking way to set the effectiveness level of the AI by changing the intervals for individual commands after which the next move can be made.

Selection of the commands follows this process:

1. If sheep are scattered, issue "Look Back".
2. If all sheep are in one group find and periodically update a path from this group to the next checkpoint.
3. Use the angle between dog, sheep's centroid and next waypoint on the path to issue a circling command
4. Issue "Walk Up" when the angle between dog, centroid of sheep and next waypoint of path is close to 180°.
5. If the dog is walking up and the angle has changed beyond a tolerance (either due to having reached the next waypoint or noise) go to 3.
6. If the angle has improved, issue "walk up" again.

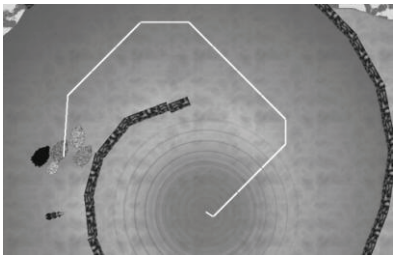

Figure 5: Shepherd steering path towards a gathering checkpoint

To simulate some foresight about the likelihood of sheep becoming stuck at obstacles, the steering path avoids proximity to squares blocked by walls or water, by making use of a pre-calculated influence map used in a similar way as the sheep influence map used for the dog when avoiding proximity to sheep (see Figure 5). To make sure that sheep will actually pass through gate checkpoints, an extra point after the checkpoint on a line parallel to the checkpoints normal is added to the path and the path is not smoothed, as a few additional "away" and "come-bye" commands give the shepherd a more human feel.

Performance of AI Shepherd

Looking at the results of AI players it becomes clear that the player performs quite consistently on terrains which are relatively spacious and simple, and AI players will finish any of the tested levels (see Figure 6 & Table 1) in a reasonable time. The AI starts to produce more unpredictable results on maps which either have very narrow passes such as level 3 and 5, or which contain a large number of sheep like level 4. This is believed to be due to three main causes. The way in which the steering path is currently calculated does not produce optimal results if there are not at least three squares between two obstacles that need to be passed. In environments where there are only one or two squares in between there is no safer square for the path search to choose, resulting in waypoints which can be dangerously close to walls, and often a split flock.

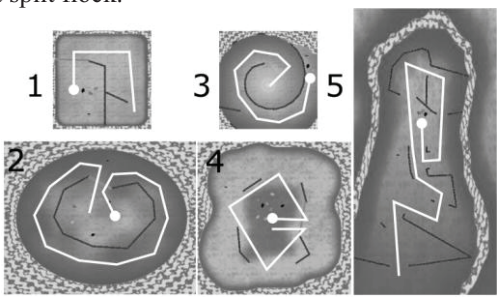

Figure 6: Test levels and approximate level flow (start marked by circle)

Measured times in seconds for AI Players				AI average deviation from mean time		Hum. times
Lvl	Min	Max	Avg	sec	sec/Avg	Avg
1	38	51	44	2.5	0.057	46
2	128	185	153	12.9	0.084	192
3	50	156	75	15.7	0.21	65
4	146	269	189	35.8	0.189	195
5	133	302	220	50.1	0.228	209

Table 1: AI shepherd results (20 rounds per level)

Also, with the AI always issuing look back commands as soon as the flock splits, this can produce situations in which the gathering of all sheep is not beneficial for the overall result. For example, sheep which have already passed a gate checkpoint are sometimes returned to the main flock which has not yet done so. With a large flock of sheep the probability of splitting increases, and with it the probability of the AI making a wrong decision with the current rules. Lastly, it is quite challenging to guide sheep through narrow obstacles as they will perceive the walls and steer away from them, which results in sheep turning away from the wall if they are not optimally aligned beforehand. Of course this is a correct reflection of the real-life difficulty of guiding a flock of sheep through narrow and closely-spaced obstacles.

Gameplay Impressions

Many of the main aims for this project such as having realistic dog, sheep and shepherd behaviors and enjoyable gameplay are only subjectively measurable. Volunteers, most of which had at least at some point observed actual shepherding on TV or in person were asked to evaluate such factors on a scale from 1 to 6 (see table 2 below). Results from user testing showed that the gameplay was perceived as enjoyable and that the different AI components largely behaved as intended for the game

which provides a good degree of simulation, whilst still being accessible for new players who can quickly learn to predict what effects their actions will have.

Statement	Min / Average / Max
Dog perceived as intelligent	3 / 4.78 / 6
Shepherding predictable	3 / 4.44 / 6
Realistic sheep behavior	3 / 5 / 6
Gameplay enjoyable	4 / 5.11 / 6

Table 2: Assessment of subjective AI results by testers (game tested by 9 users)

Conclusions

Our game shows the power of influence mapping to create realistic behavior and enjoyable gameplay, starting from a thorough understanding of the real-world situation upon which the game is based. We have split the shepherding problem into its natural parts: sheep, shepherd and sheepdog, using influence mapping for path calculation to model realistic behavior of a sheepdog. The simple current implementation for the AI shepherd places commands which are exceeding or at least coming close to the effectiveness of a human having played the game for some time, thus providing a good amount of challenge to users.

References

Alexander, B. 2006. Flow Fields for Movement and Obstacle Avoidance. In *AI Game Programming Wisdom 3*. Charles River Media: Boston, Massachusetts

Burke, R.& Isla, D. & Downie, M. & Ivanov, B & Blumberg, Y. 2001. CreatureSmarts: The Art and Architecture of a Virtual Brain. In *Proceedings of the Game Developers Conference*, pp. 147-166, San Jose, CA

Capcom, 2009. Flock! http://www.capcom-europe.com/games/Flock-Xbox-360

Core Design, 2002. Herdy Gerdy http://www.herdygerdy.co.uk/

Frontier Development, 2003 Dog's life http://www.dogslifeps2.com/

Fu, D. & Houlette, R. 2004. The ultimate Guide to FSMs in Games. In *AI Game Programming Wisdom 2*. Charles River Media: Hingham, Massachusetts

Grandin, T. 1989. Behavioral Principles of Livestock Handling. In *Professional Animal Scientist*, December 1989 pages 1-11

Laming, B. 2008. The MARPO Methodology. In *AI Games Programming Wisdom 4* Course Technology: Boston, Massachusetts

Lien, J. & Bayazit, O. Sowell, R. & Rodriguez S, & Amato, N. 2004. Shepherding Behaviors. In *IEEE International Conference on Robotics and Autom*. (pp. 4159-4164)

Lien, J. & Rodriguez S, & Malric, J. & Amato, N. 2005. Shepherding Behaviors with Multiple Shepherds In *Proc. IEEE Int. Conf. Robot. Autom.* (ICRA), Apr 2005.

Miles, C. & Louis, S. 2006. Co-evolving real-time strategy game playing influence map trees with genetic algorithms. In *Proc. of the Congress on Evolutionary Computation*. Vancouver, Canada

Molyneux, P. 2001. Postmortem: Lionhead Studios' Black & White. http://www.gamasutra.com/view/feature/3067/postmortem_lionhead_studios_.php

Nintendo. 2006. The Legend of Zelda: Twilight Princess http://www.zelda.com/tp/

Paanakker, F. 2008. Risk-Adverse Pathfinding Using Influence Maps. In *AI Games Programming Wisdom 4* Course Technology: Boston, Massachusetts

Phipps Associates, 1982. Robot Chase http://www.worldofspectrum.org/infoseekid.cgi?id=0016021

Raine, L. 2001. Words and Whistles: Stockdog Commands. http://www.meekersheepdog.com/event.htm

Reynolds, C. 1987 Flocks, Herds, and Schools: A Distributed Behavioral Model. In *Computer Graphics,* 21(4), July 1987, pp. 25-34.

Reynolds, C. 2000. Interaction with groups of autonomous characters. In Game Developer's conference 2000

Shulaw, W. 2005. ASI Sheep Care Guide 2005 Edition

Touzour, P. 2004 Search Space Representations. In *AI Game Programming Wisdom 2* Charles River Media: Hingham, Massachusetts

Vaughan, Richard & Sumpter, Neil & Frost, Andy & Cameron, Stephen 1998. Robot Sheepdog Project achieves automatic Flock Control In *Proc. Fifth International Conference on the Simulation of Adaptive Behaviour*

Yiskis, E. 2004. A Subsumption Architecture for Character-based Games. In *AI Game Programming Wisdom 2* Charles River Media: Hingham, Massachusetts