Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Rok Šeme

Računalniška simulacija črede ovc in psa ovčarja

DIPLOMSKO DELO
UNIVERZITETNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

izr. prof. dr. Iztok Lebar Bajec
MENTOR

Ljubljana, 2016

Tematika naloge:

V diplomski nalogi preučite vir Ginelli, F., et al. (2015), Intermittent collective dynamics emerge from conflicting imperatives in sheep herds, doi: 10.1073/pnas.1503749112 *ter* Strömbom, D., et al. (2014), Solving the shepherding problem: heuristics for herding autonomous, interacting agents, doi: 10.1098/rsif.2014.0719. *Predstavljena algoritma obnašanja ovc in psa ovčarja najprej poustvarite, nato pa poizkusite izboljšati delovanje algoritmov. Primerjajte uspešnost obeh algoritmov in rezultate smiselno ovrednotite.*

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani izjavljam, da sem avtor dela, da slednje ne vsebuje materiala, ki bi ga kdorkoli predhodno že objavil ali oddal v obravnavo za pridobitev naziva na univerzi ali drugem visokošolskem zavodu, razen v primerih kjer so navedeni viri.
S svojim podpisom zagotavljam, da:

- sem delo izdelal samostojno pod mentorstvom izr. prof. dr. Iztoka Lebarja Bajca,
- so elektronska oblika dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko in
- soglašam z javno objavo elektronske oblike dela v zbirki "Dela FRI".

— Rok Šeme, Ljubljana, september 2016.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Rok Šeme

Računalniška simulacija črede ovc in psa ovčarja

POVZETEK

Računalniške simulacije obnašanja se danes uporabljajo skoraj povsod: igre, filmi, robotika, itd. Namen razvoja modelov obnašanja je, da bi čim bolj poustvarili naravno vedenje živali, kot so ptice, ribe, žuželke, ovce. Namen diplomske naloge je preučiti aktualne modele obnašanja ovc in psa ovčarja. V nalogi je predstavljena implementacija dveh modelov obnašanja ovc in en model obnašanja psa. Algoritmi so povzeti po dveh člankih *Strömbom, D., et al. (2014), Solving the shepherding problem: heuristics for herding autonomous, interacting agents, doi: 10.1098/rsif.2014.0719* ter *Ginelli, F., et al. (2015), Intermittent collective dynamics emerge from conflicting imperatives in sheep herds, doi: 10.1073/pnas.1503749112*, ki sta bila glavno vodilo pri ustvarjanju simulacije, nato pa je bilo dodanih nekaj izboljšav. Narejeni so bili testi, pri katerih se je ugotavljala uspešnost modela ovčarja pri obeh modelih ovc, pri različno velikih čredah.

Ključne besede: simulacija, ovca, ovčar, čreda, model obnašanja, skupinsko vedenje

University of Ljubljana
Faculty of Computer and Information Science

Rok Šeme

Computer simulation of a sheep herd and a sheepdog

ABSTRACT

Computer simulations of the behaviour are nowadays used almost everywhere: games, movies, robotics, etc. The purpose of developing models of behaviour is to recreate as closely as possible the natural behaviour of animals, such as birds, fish, insects, sheep. The purpose of the thesis was to examine the latest models of the behaviour of sheep and the sheepdog. The paper presents the implementation of two sheep behaviour models and one dog behaviour model. Algorithms are summarized according to two articles *Strömbom, D., et al. (2014), Solving the shepherding problem: heuristics for herding autonomous, interacting agents, doi: 10.1098/rsif.2014.0719* and *Ginelli, F., et al. (2015), Intermittent collective dynamics emerge from conflicting imperatives in sheep herds, doi: 10.1073/pnas.1503749112*, both of which were as the main guidelines in the creation of the simulation, and then there were added a few improvements. Tests have been made, in which we observed the success of the sheepdog model with both sheep models, at the different sized herds.

Key words: simulation, sheep, sheepdog, herd, behaviour model, collective animal behaviour

KAZALO

1 Uvod

Ovčereja je približno 5000 let prisotna pri ljudeh, saj od ovc lahko pridobivamo mleko, meso in volno[1]. V Evropi je bila v sredini 19. stoletja v velikem razmahu, nato pa je, zaradi konkurence bombaža in sintetičnih materialov, začela upadati[2]. Naloga pastirja je da, s pomočjo ovčarja, skrbi za varnost ovc ter jih vodi od enega pašnika do drugega. To najlažje stori, če se ovce držijo v čredi. Ker pa je čreda običajno sestavljena iz velikega števila ovc, je samemu pastirju težko voditi čredo. V ta namen je človek vzgojil psa, da mu pri temu pomaga. Zaradi zanimivosti področja, se znanstveniki s čredami ukvarjajo že zelo dolgo. Področje je uporabno za raziskave pri matematiki ter računalniškem modeliranju. Vendar pa še vedno ni popolnoma raziskano, saj je težko najti algoritem, ki bi natančno poustvaril obnašanje ovce in ovčarja. Težko je določiti vse dejavnike, ki vplivajo na odločitve ovc in ovčarskih psov. Zato znanstveniki delajo poskuse, v katerih opazujejo ovce in pse ter poskušajo ustvariti algoritem, ki bi čim bolj poustvaril njihovo

[1] 'Shepherd, Wikipedia', [Online] Dosegljivo: https://en.wikipedia.org/wiki/Shepherd [Dostopano: Avgust 2016]

[2] 'Ovce, Pomen reje', [Online] Dosegljivo: http://www.ovce.si/pomen-reje-ovc [Dostopano:Avgust 2016]

obnašanje. Tako je recimo avtor doktorske dizertacije [1] razvil dva močna algoritma za načrtovanje gibanja črede. Naloga vključuje manjšo skupino robotov, ki imajo interakcijo z večjo skupino vodljivih agentov, ter robota, katerega naloga je spremljanje opazovane skupine. V tem primeru je naloga težja, saj so na polju, po katerem se agenti gibajo, tudi ovire in te ne smejo ovirati robota, katerega naloga je opazovanje obnašanja. Algoritma sta kreirana tako, da efektivno vzorčita podatke in s tem omogočita hitro nadaljnjo planiranje premika. Zanimivo je tudi področje, kjer se v simulacijo vključi več psov [2]. Pri tem moramo vključiti tudi dodatni odnos, poleg odnosa ovce z drugo ovco in odnosa ovce do psa, ki je odnos psa do drugega psa. Psi morajo sodelovati skupaj, da uspešno vodijo čredo, pri tem pa je njihova komunikacija omejena samo na pozicijo in smer gibanja.

Čredni algoritmi se uporabljajo v industriji iger, filmski industriji [3] ter robotiki [4]. Lahko se jih uporabi v različne praktične namene, saj bi npr. lahko roboti pomagali ovčarjem ali pa jih celo nadomestili [5]. Roboti bi lahko pomagali ljudem v nevarnosti (npr. potres) [6] ali v primeru slabe vidljivosti. V takšnih situacijah bi imel robot nalogo, da poišče ljudi v nevarnosti in jih pripelje do varne točke, če je to možno. V nasprotnem primeru pa bi vodil reševalce do poškodovanca. Prav tako bi jih lahko uporabili za pomoč pri razlitju nafte [7]. Njihova glavna naloga bi bila preprečevanje širjenja razlitja na večje območje ter nato tudi čiščenje. Usmerjali bi lahko jate ptic, ki se približujejo letališču [8]. Ideja izhaja iz problema, s katerim se upravljalci letališč redno srečujejo. Letalo, ki vzleta ali pristaja na letališču, leti dovolj nizko, da križa svojo pot s potjo ptic. Ker se letalo ne more izogniti jati ptic, ki prileti v zračni prostor letališča, pride do trka in posledično do poškodb na letalskih motorjih. Rešitev je brezpilotno letalo, katerega naloga bi bila, da jato ptic, ki se približuje zračnemu prostoru letališča, preusmeri do varne točke.

V seminarju [9] je avtor primerjal različne algoritme obnašanja psov pri vodenju ovc. Preučil je nekaj modelov obnašanja psa, ki jih je nato implementiral in jih primerjal po uspešnosti. Pri testiranju je uporabil dve merili za ugotavljanje uspešnosti. Za prvo merilo je uporabil merjenje povprečnega časa, ki ga je potreboval ovčar za uspešno končano nalogo. Kot drugo merilo je vzel povprečno razdaljo poti, ki jo je moral opraviti ovčar. Dve leti kasneje je v študiji [10], v kateri je sodeloval kot soavtor, ponovil testiranja z novim modelom obnašanja psa. Ta model so avtorji razvili na podlagi ukazov, ki jih

uporabljajo pastirji pri psu. Pri testiranju so ugotovili, da je model, ki so ga razvili, nekoliko boljši glede na povprečni čas, kot pa pri ostalih modelih psa. Vendar pa je njegova uspešnost nižja, če se testiranju doda časovna omejitev.

V diplomski nalogi sem opravil primerjavo uspešnosti ovčarja pri različnih modelih obnašanja ovc. Najprej je predstavljen generalni model obnašanja, nato pa je opisana implementacija algoritmov modelov, po člankih [11, 12], ki jih poskušamo poustvariti ter nekaj dodanih izboljšav. Sledi opis testiranja in analiza pridobljenih rezultatov. Navedenih je tudi nekaj možnih nadgradenj.

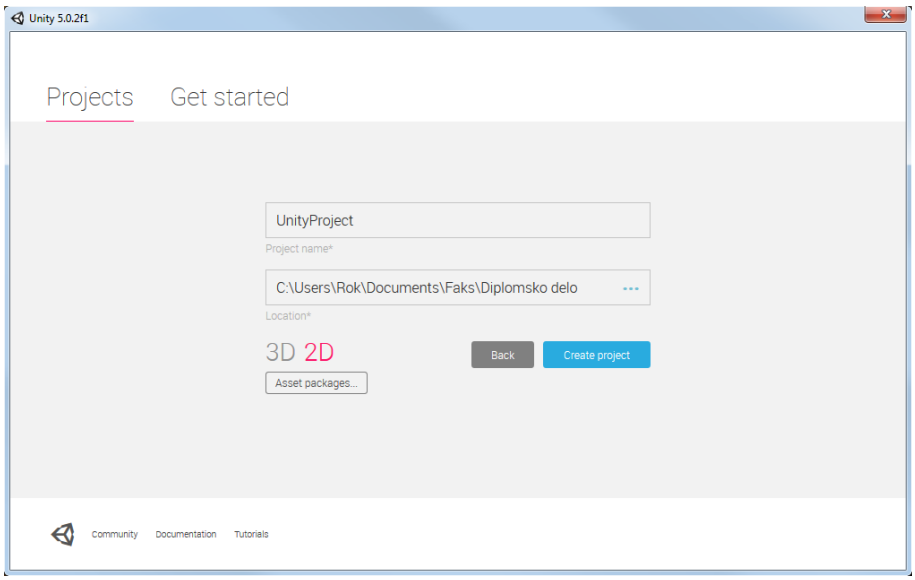
2 Uporabljena orodja in tehnologije

Unity Technologies je razvil pogon za razvoj iger[1], ki deluje na različnih platformah in zato z njim enostavno razvijamo projekte, tako 2D kot 3D. Te brez problema optimiziramo in jim izboljšamo izgled. Skozi razvoj so razvijalci tudi dosegli, da iz projekta z lahkoto ustvarimo končni izdelek za več naprav. Unity je enostaven za uporabo in omogoča, da si ga uporabnik prilagodi za svoje potrebe. Omogoča izbiro med dvema programskima jezikoma in sicer med *C#* in *Javascript*.

2.1 Primer 2D projekta

Ob zagonu programa *Unity*, se odpre okno, ki nam ponudi obstoječe projekte, na katerih smo že delali, ali pa kreiranje novega projekta. Pri izbiri slednjega se odpre podokno, v katerem z lahkoto ustvarimo novi projekt, kot prikazuje slika 2.1. Po želji poimenujemo projekt, izberemo kje želimo, da nam ga *Unity* ustvari, izberemo tip projekta 2D in kliknemo na "Create project".

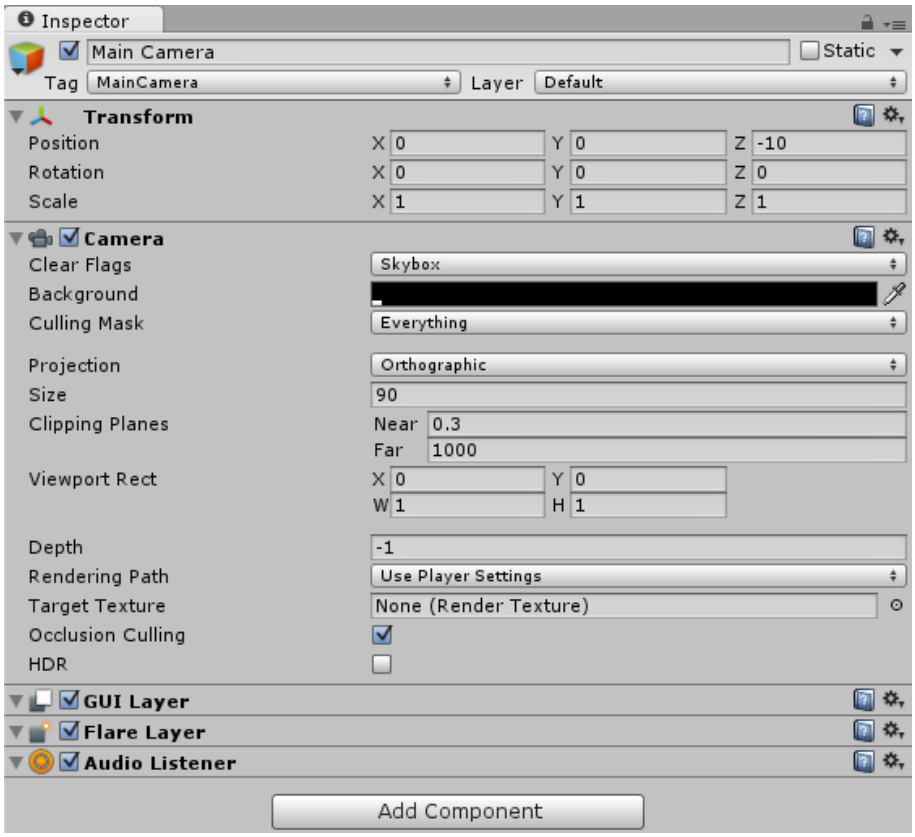
[1] 'Unity', [Online] Dosegljivo: https://unity3d.com/unity [Dostopano: Avgust 2016]



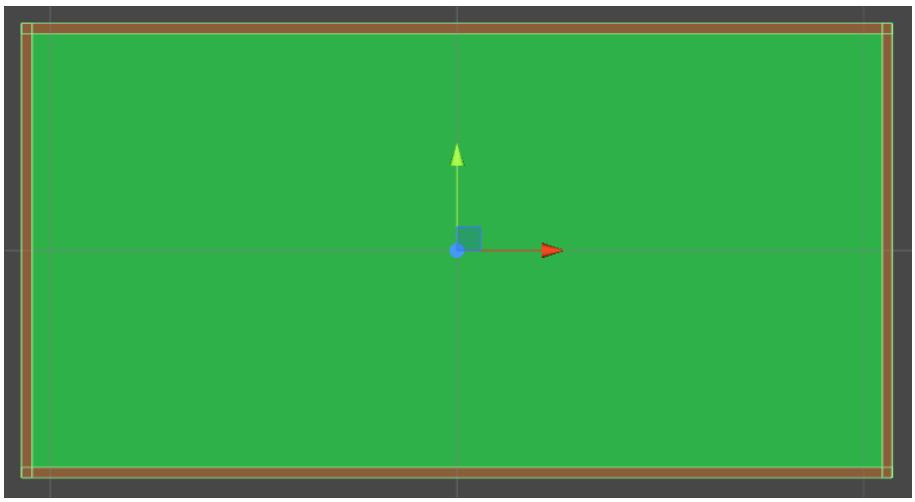
Slika 2.1 Okno za izdelavo *Unity* projekta.

Najprej prilagodimo kamero za naše potrebe. To storimo tako, da izberemo glavno kamero v hierarhiji in v seznamu komponent spreminjamo željene parametre, primer slika 2.2. Priporočljivo je, da je kamera odmaknjena od ravnine, ki gre skozi koordinatno središče. Popravimo velikost polja, za katerega želimo, da ga kamera zajame, spremenimo barvo ozadja v črno, ...

V projekt moramo dodati podlogo, na kateri se bo vse dogajalo. Podlogo lahko ustvarimo v poljubnem programu za risanje ali oblikovanje slik, lahko pa jo tudi prenesemo iz interneta. Izbrano podlogo shranimo v mapo projekta *Assets*, da lahko do nje dostopamo iz *Unity*-a. V seznamu komponent popravimo parametre podloge, tako da nam ustrezajo. Podlogo nato dodamo na sceno. To storimo tako, da jo povlečemo v prostor in ji popravimo pozicijo v seznamu komponent. Vendar, pa je podloga trenutno samo slika. Ni del fizičnega sveta, kar pomeni, da ostali elementi v sceni ne morejo imeti interakcije z njo. Ker objektov ne omejuje, lahko prečkajo njene meje. To popravimo s komponento za zaznavanje trkov, ki jo dodamo na vsak zid ali oviro na naši podlogi. V seznamu komponent kliknemo "Add Component" in pod "Physics 2D" izberemo nam najbolj ustrezno komponento za zaznavanje trkov. Ker pa *Unity* center izbrane komponente za zaznavanje trkov vedno postavi v center objekta in poskuša prilagoditi njegovo velikost glede



Slika 2.2 Prilagajanje nastavitev kamere v seznamu komponent.



Slika 2.3 Podloga z dodanimi komponentami za zaznavanje trkov (okvir svetlo zelene barve) na njenih robovih.

na velikost objekta, ga je potrebno dodatno prilagoditi. Po navadi samo ena komponenta za zaznavanje trkov ni dovolj za popolno delovanje podlage, zato jih je potrebno dodati več in vsakega postaviti na ustrezno mesto ter mu prilagoditi velikost. Primer je slika 2.3, kjer je podloga pašnik z ograjo in na vsako stranico ograje je dodana komponenta za zaznavanje trkov v pravokotni obliki.

Po istem postopku lahko dodamo ostale objekte na sceno projekta ali pa v meniju pod objekti izberemo preprosto 2D platno. V obeh primerih pa ne smemo pozabiti na dodajanje komponente za zaznavanje trkov, saj brez nje objekt ne bo mogel imeti interakcije z drugimi objekti. Če pa bomo objekt premikali po sceni, je priporočeno, da mu dodamo tudi komponento togega telesa.

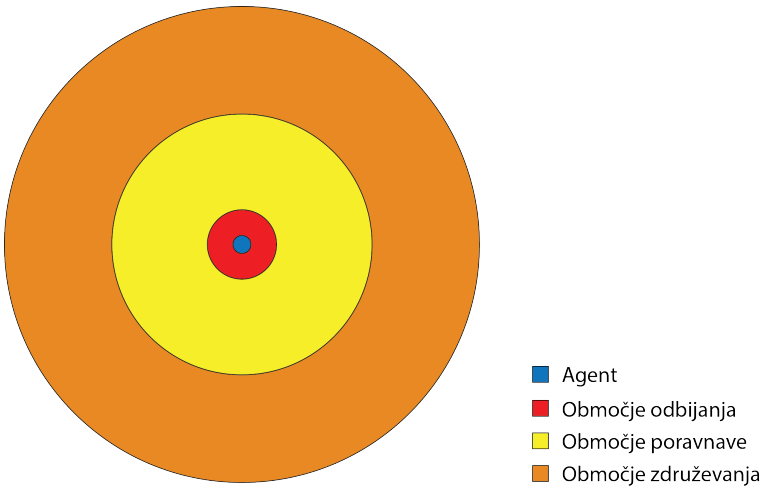
Objektu, ki mu želimo definirati obnašanje, lahko dodamo tudi skripto. Na voljo imamo dva programska jezika, to sta *Javascript* in *C#*. Ob kreiranju skripte, nam *Unity* sam ustvari dve najbolj pomembni funkciji, to sta `Start()` in `Update()`. Funkcija `Start()` se kliče ob generiranju objekta in je namenjena za nastavitev parametrov. Funkcija `Update()` pa nam služi za sprotno popravljanje parametrov, časovnikov, premikanje in zajem vhodnih podatkov. Ta funkcija se kliče za vsako novo sliko, kar pomeni, da se časovni interval klica med posameznimi klici lahko razlikuje. Če želimo imeti enakomerne časovne intervale, moramo uporabiti funkcijo `FixedUpdate()`, ki se kliče za vsak korak izračuna

fizike. Uporaba te funkcije je priporočljiva, ko želimo upravljati objekte, na katere deluje fizika. Zelo uporabna je tudi funkcija `OnCollisionEnter2D (Collision2D)` ali pa `OnCollisionEnter3D (Collision3D)`, če delamo v tri-dimenzionalnem prostoru. Ti funkciji se sprožita, ko pride do trka, in v njima lahko dostopamo do objektov pri katerih je prišlo do trka. Na primer izpis 2.1 je preprosta skripta, kjer sta v funkciji `Start()` definirana vektorja smeri za levo in desno in določena primarna smer. V funkciji `Update()` je koda za premik objekta v trenutni smeri. To storimo tako, da izračunamo pot, ki jo objekt opravi med posameznim intervalom klica, in objekt za toliko tudi premaknemo. Dodana je tudi funkcija, ki se proži ob trku, v kateri preprosto zamenjamo smer premikanja. Rezultat celotne skripte je objekt, ki se premika po polju od ene strani do druge.

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class MoveObject : MonoBehaviour {
5
6          float speed = 10.0f;
7          float directionLeft;
8          float directionRight;
9          float currentDirection;
10         bool goingLeft;
11
12         // Use this for initialization
13         void Start () {
14                 directionLeft = Mathf.Atan2 (0.0f, -100.0f);
15                 directionRight = Mathf.Atan2 (0.0f, 100.0f);
16                 currentDirection = directionLeft;
17                 goingLeft = true;
18         }
19
20         // Update is called once per frame
21         void Update () {
22                 Vector3 direction = new Vector3(Mathf.Cos(currentDirection)
                       , Mathf.Sin(currentDirection), 0.0f);
23                 Vector3 step = transform.position + Time.deltaTime * speed
                       * direction;
24                 Vector3 p = Vector3.MoveTowards(transform.position, step,
```

```
                    speed);
25                  GetComponent<Rigidbody2D>().MovePosition(p);
26          }
27
28          void OnCollisionEnter2D(Collision2D col)
29          {
30                  if(goingLeft) {
31                          goingLeft = false;
32                          currentDirection = directionRight;
33                  } else {
34                          goingLeft = true;
35                          currentDirection = directionLeft;
36                  }
37          }
38
39  }
```

Izpis 2.1 Primer skripte v *C#*.



Slika 2.4 Primer velikosti območij v modelu Boids.

2.2 Modeli obnašanja

Pri izdelavi modela obnašanja skupine se za osnovo najpogosteje vzame model Boids [3], ki ima tri območja okoli vsakega aktivnega agenta. Vsako od teh območij ima svoj namen in se po njemu tudi imenuje. Osnovne tri težnje, ki naj bi jih imel vsak agent so: a) odmikanje od agentov, ki so preblizu, b) poravnava smeri premikanja z agenti v bližnji okolici ter c) združevanje v skupino. Območje odbijanja je najmanjše območje, saj preprečuje trke med agenti in tako je potrebno opazovati samo agente, ki so preblizu. Naslednje območje po velikosti je območje poravnave. To območje je srednje velikosti, ker namreč živali v naravi popravljajo svojo smer premikanja samo glede na sosede in ignorirajo tiste, ki so preveč oddaljeni. Največje je območje združevanja. To območje predstavlja celotno vidno ali zaznavno polje agenta in služi za zaznavo lokalnega centra skupine. Agent teži proti tej točki, ker se s tem ustvarja skupina agentov. Primer velikosti teh območij predstavlja slika 2.4.

Območje odbijanja, kjer se preprečujejo trki med agenti, je običajno izvedeno tako, da agent dobi silo, ki je usmerjena stran od vseh zaznanih agentov v tem območju. Ta sila je eden izmed oteženih smernih vektorjev, ki jih seštejemo pri izračunu končne smeri. Naslednjo silo, ki deluje na agenta, dobimo v območju poravnave. V tem območju vzamemo smerne vektorje vseh zaznanih agentov (sosedov) in iz njih izračunamo njihovo

skupno (povprečno) smer premikanja. To je druga sila, ki deluje na agenta. Iz vseh zaznanih agentov, ki se nahajajo v območju približevanja, se izračuna njihov lokalni center in nato silo, ki obravnavanega agenta vleče proti tej točki.

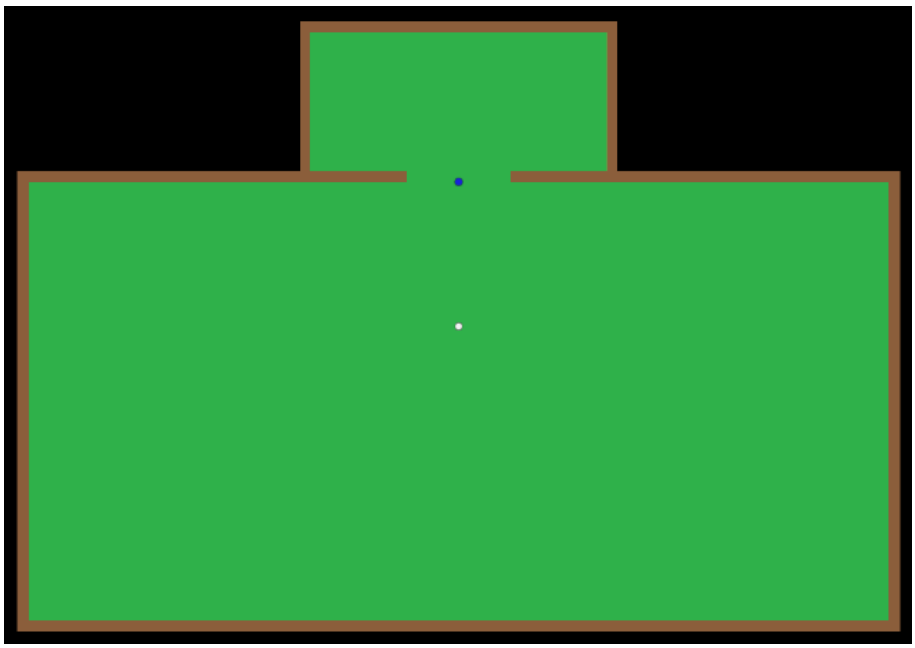
Velikosti teh območij se definira glede na tip živali, ki jih želimo simulirati. Prav tako so od tega odvisne tudi moči sil v posameznem območju. Model obnašanja se lahko nadgradi s tem, da mu dodamo še dodatne lastnosti. To je lahko strah pred grožnjo, ki usmeri agenta v nasprotno smer in mu omogoči beg pred plenilcem [11]. Pri simuliranju obnašanja ptic lahko dodamo ciljno usmerjenost ter s tem ponazorimo selitev ptic, ko se orientirajo po elektromagnetnih čutilih. Ptice pa lahko iščejo tudi vizualne orientacijske znake, kot je na primer sonce [13]. Dodamo lahko tudi silo sledenja, kjer je v skupini določen vodja, ki mu sledijo vsi ostali agenti.

Model Boids je v osnovi namenjen za 3D simulacije, saj ko delamo recimo z jato ptic ali rib, za nazorno simulacijo potrebujemo vse tri dimenzije. Za simuliranje ovc, ki se premikajo le po pašniku, je dovolj da imamo samo 2D prostor, saj nikoli ne pride do situacije, da bi bila ena ovca nad drugo. Težava lahko nastane, če so agenti preveč narazen, da bi se med seboj zaznali, in se zaradi tega ne morejo združiti v čredo. Vendar pa se to v mojem primeru ne zgodi pogosto, saj to preprečujem s tem, da je vedno prisoten tudi ovčarski pes, ki zgubljene ovce prežene nazaj v čredo.

3 Implementacija modelov obnašanja

Podloga oziroma okolje, v katerem se bodo gibali agenti, je sestavljeno iz dveh delov. Večje polje predstavlja pašnik, na katerem se ovce prosto pasejo in je začetno polje. Manjše polje pa predstavlja ograjo ali hlev, kamor mora ovčar ovce prignati. To predstavlja ciljno polje. Polji sta drug ob drugem in med njima je možen prehod skozi luknjo v ogradi, kot prikazuje slika 3.1, kjer je začetno polje velikosti 207×104 enot, ciljno polje velikosti 70×32 enot ter prehod med njima s širino 23.5 enot. V polje sta postavljena tudi preprosta objekta, ki nam predstavljata ovco (bela pika s premerom 1 enote) in ovčarja (modra pika s premerom 1.2 enote).

Podloga ima dodano skripto, ki skrbi za ustrezno število elementov na polju. Nanjo je vezan osnovni objekt, to je ovca, in iz njega naredi ustrezno število kopij ter jih naključno razporedi po polju. Skripta tudi vključuje kodo za merjenje časa, ki ga izpiše ob uspešno zaključeni simulaciji.



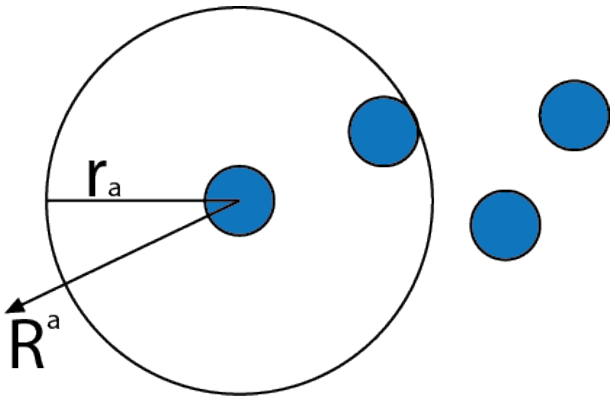
Slika 3.1 Osnovna postavitev okolja.

3.1 Model obnašanja ovc po Strömbom in sod.

Model obnašanja ovc, ki so ga predstavili Strömbom in sod. [11] je dokaj preprost model, saj se predvsem osredotoča na odnos ovce in psa. Obnašanje ovce je narejeno po prilagojenem modelu Boids [3]. Prvotno je model Boids sestavljen iz treh območij okoli agenta: območje odbijanja, območje približevanja in območje poravnave. Vendar pa lahko pri modelu po Strömbom in sod., zanemarimo območje poravnave, saj se ovce skupinsko premikajo v isto smer, samo ko so ogrožene. Namreč, ker se želijo vse ovce odmakniti stran od grožnje, pri tem pa jih še zmeraj skupaj drži težnja združevanja, nam to predstavlja smerni vektor gibanja ovc v isto smer. Dodano pa je območje strahu. Namen tega območja je, da ovca zazna grožnjo znotraj tega območja in se poskuša odmakniti stran od nje.

3.1.1 Izogibanje trkom med agenti

V naravi ovce držijo med seboj nekaj razdalje, tako da ima vsaka ovca dovolj manevrskega prostora. V primeru, da je v bližini grožnja, se ta razdalja tudi zmanjša ali pa se lahko celo zgodi, če je grožnja zelo blizu oziroma je malo prostora, da se ovce v čredi dotikajo



Slika 3.2 Območje odbijanja in odbojna sila.

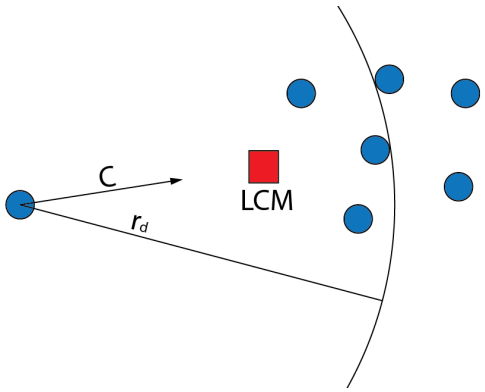
druga druge. Za poenostavitev modela obnašanja, delamo samo z eno razdaljo, to je r_a. Pri vsakem agentu preverimo ali ima v svojem območju odbijanja, ki je velikosti r_a, enega ali več agentov. Za vse najdene agente izračunamo smerni vektor $\mathbf{R}_i^a$, ki je usmerjen v nasprotno smer posamičnega sosednjega agenta, kot je prikazano na sliki 3.2. Ovce tudi lahko pridejo druga do druge, če so ostale sile, ki tudi delujejo na posamezno ovco, dovolj močne, da premagajo odbojno silo.

Smerni vektor $\mathbf{R}_i^a$ izračunamo na sledeči način, če ima i-ti agent k sosedov, v oddaljenosti manj kot r_a in se ti nahajajo na pozicijah $\mathbf{A}_1, \ldots, \mathbf{A}_k$, je odbojna sila na i-tega agenta definirana po enačbi 3.1. Bolj preprosto, poziciji i-tega agenta odštejemo pozicijo j-tega agenta, s tem dobimo vektor, ki je obrnjen stran od j-tega agenta. Ta vektor nato normaliziramo in ga dodamo vsoti vektorjev odbojne sile.

$$\mathbf{R}_i^a = \sum_{j=1}^{k} \frac{1}{||\mathbf{A}_i - \mathbf{A}_j||}(\mathbf{A}_i - \mathbf{A}_j) \tag{3.1}$$

3.1.2 Združevanje agentov v čredo

Ovce se združujejo v čredo, saj ima tako posamezna ovca več možnosti za preživetje, če je čreda napadena. Zato na vsako ovco deluje tudi čredni nagon. Bistvo je, da se ovca približuje centru črede. Ker pa ovca v naravnem svetu ne pozna lokacij vseh ovc, razen tistih, ki so ji blizu, pri računanju centra črede ne izračunamo globalnega težišča (GCM), ampak lokalno težišče (LCM) njenih sosedov. V modelu obnašanja po članku se uporablja za računanje LCM-ja k najbližjih sosedov, ne glede na njihovo oddaljenost.



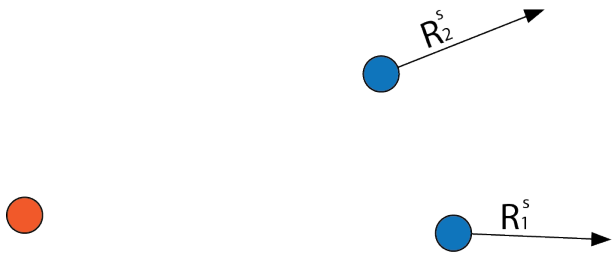
Slika 3.3 Iskanje LCM.

To pomeni, da se lahko zgodi, da je neka ovca izbrana med k najbližjih ovc, istočasno pa oddaljena toliko, da v realnem svetu ne bi bila v vidnem polju. Zato sem model obnašanja nadgradil s tem, da za računanje LCM uporabim samo ovce, ki so v radiju vidnega polja, kot prikazuje slika 3.3. S tem sem tudi spremenil parameter k v spremenljivko, ki je sedaj odvisna od števila ovc, ki so v bližini. Zaradi tega lahko spremenljivka k doseže tudi vrednost števila vseh ovc, ki so na polju in se ob računanju LCM tako izračuna GCM. Možen je tudi obraten primer, kjer agent v svojem radiju zaznavanja nima nobenega drugega agenta. V tem primeru je LCM enak kar poziciji agenta samega.

Če ima i-ti agent k sosedov, v oddaljenosti manj kot r_d in se ti nahajajo na pozicijah $\mathbf{A}_1, \ldots, \mathbf{A}_k$, je lokalno težišče (LCM) za agenta i izračunano po enačbi 3.2. Bolj preprosto, poziciji i-tega agenta prištejemo pozicije njegovih sosedov na razdalji r_d. To vsoto nato delimo s številom sosedov, povečanim za 1, saj pri izračunu uporabimo tudi opazovanega agenta. Tako dobimo povprečno pozicijo vseh okoliških agentov, kar je enako LCM. Da dobimo smerni vektor $\mathbf{C}_i$, ki kaže proti LCM, pozicijo agenta odštejemo od LCM, kot prikazuje enačba 3.3. Tako izračunamo smer sile združevanja.

$$\mathbf{LCM}_i = \frac{\mathbf{A}_i + \sum_{j=1}^{k} \mathbf{A}_j}{k+1} \tag{3.2}$$

$$\mathbf{C}_i = \mathbf{LCM}_i - \mathbf{A}_i \tag{3.3}$$



Slika 3.4 Odmikanje agentov od grožnje.

3.1.3 Odmikanje agentov od ovčarja

V simulacijskem okolju se giba tudi ovčar. Od njega se ovce odmikajo, saj jim le ta predstavlja grožnjo. Ko se ovčar ovci približa na razdaljo, ki je krajša od razdalje r_s, na ovco začne delovati odbojna sila ovčarja, ki kaže v obratni smeri najkrajše poti do njega, kot prikazuje slika 3.4. Smer odbojne sile ovčarja ni odvisna od sosednih ovc. Pri izračunu se opazuje samo pozicija grožnje in opazovanega agenta. Smerni vektor odbojne sile ovčarja $\mathbf{R}_i^s$, ki kaže direktno stran od ovčarja, dobimo tako, da odštejemo pozicijo ovčarja poziciji ovce po enačbi: $\mathbf{R}_i^s = \mathbf{A}_i - \mathbf{S}$, kjer je $\mathbf{S}$ pozicija ovčarja, $\mathbf{A}_i$ pa pozicija opazovane ovce.

3.1.4 Določitev končnega smernega vektorja

Smer premikanja $\mathbf{H'}_i$, v katero bo šel agent v naslednjem časovnem koraku, je linearna kombinacija vseh normaliziranih smernih sil, vztrajnosti prejšnjega koraka $\mathbf{H}_i$ in naključnega vektorja ε: $\mathbf{H'}_i = h\hat{\mathbf{H}}_i + c\hat{\mathbf{C}}_i + p_a\hat{\mathbf{R}}_i^a + p_s\hat{\mathbf{R}}_i^s + e\hat{\varepsilon}_i$, pri čemer so uteži izbrane tako, da drži pogoj $p_a > c > p_s > h > e$. Razlog za to neenakost je, da mora biti odbojna sila med agenti dominantna, saj se le tako ohranja velikost črede. V naravi se ovce tudi raje združujejo v gručo, kot pa da se razkropijo ob prisotnosti psa. Iz tega lahko sklepamo, da mora imeti sila združevanja večjo težo, kot pa sila odmikanja od ovčarja. Vztrajnost prejšnjega koraka je vključena izključno z namenom, da se prepreči ostre zavoje v premikanju agenta. S tem dobimo bolj gladko in naravno premikanje. Zaradi tega je dovolj, da je podrejena vsem ostalim interakcijam. Namen naključnega vektorja je, da naredi šum v premikanju agenta, saj se v naravi ovce ne premikajo po idealni liniji. Poleg tega pa nam ta vektor tudi kot rešitev, če se agent znajde v mrtvi situaciji, na primer, če se ujame v kot. V primeru, da agent ni ogrožen, pa nam služi za

Tabela 3.1 Seznam uporabljenih parametrov v simulaciji z njihovimi vrednostmi za model obnašanja po Strömbom in sod. [11].

Parameter	Opis	Uporabljena vrednost
r_d	Območje zaznave sosednjih agentov	15
r_s	Območje zaznave ovčarja	30
r_d	Območje preprečevanja trkov	2
p_a	Relativna moč odbijanja med agenti	2
c	Relativna moč privlačne sile v čredo	1.05
p_s	Relativna moč odbojne sile od ovčarja	1
h	Relativna moč nadaljevanja v isti smeri	0.5
e	Relativna moč šuma	0.3
p	Verjetnost za nastanek šuma	0.05
s	Hitrost premikanja	5

simulacijo pasenja. Šum ni stalno prisoten, ampak se pojavi z verjetnostjo p.

Agent se bo v naslednjem časovnem koraku premaknil v smeri $\hat{\mathbf{H}}'_i$ s hitrostjo v. Bolj preprosto, normaliziran smerni vektor $\hat{\mathbf{H}}'_i$ množimo s hitrostjo premika in velikostjo časovnega koraka. To storimo zato, da dobimo pravilno velik smerni vektor. Ta novi vektor nato prištejemo lokaciji agenta ter s tem dobimo njegovo novo pozicijo:

$$\mathbf{A}'_i = \mathbf{A}_i + \Delta t v \hat{\mathbf{H}}'_i.$$

V tabeli 3.1 so navedeni vsi parametri, ki jih potrebujemo pri izračunih, ter njihove vrednosti, ki smo jih uporabili v simulaciji.

3.2 Model obnašanja ovc po Ginelli in sod.

Model obnašanja ovc po Ginelli in sod. [12] je bolj kompleksen kot model obnašanja ovc po Strömbom in sod. [11] oziroma prvotnega modela Boids. Pri razvoju tega modela so raziskovalci opazovali veliko čredo ovc na ograjenem ravnem terenu. Opazili so, da so se ovce kmalu po prihodu na pašnik začele počasi odmikati ena od druge. Nato je sledilo nenadno združevanje v skupino, pri čemer pa ni bil povzročitelj zunanji dejavnik, na primer zvočni signal ali grožnja. Po izoblikovanju črede, so ovce ponovno začele povečevati raz-

daljo med seboj ter čez čas je sledilo ponovno nenadno združevanje v čredo. To zaporedje dogodkov se je ponavljalo znova in znova. Iz tega so raziskovalci sklepali, da čreda ovc med pašo zadovoljuje dve težnji: (I) povečevanje osebnega prostora posamezne ovce in (II) varovanje pred plenilci v veliki skupini. Prva (I) služi za povečevanje raziskanega prostora, ki ga opravi posameznik. S tem čreda doseže lažje premikanje po pašniku z velikim pokritjem terena. Z drugo (II) čreda ovc doseže, da ima ob primeru napada, posameznik večje možnosti za preživetje.

Ostali modeli ne morejo upodobiti kompleksnosti obnašanja, ki ga je zaznala raziskovalna skupina. Predvidevajo namreč, da so interakcije med posamezniki lokalne in so lahko izražene s kombinacijo poravnav, privlačnih ter odbojnih sil. Če je privlačna sila prešibka, skupina živali ne more obdržati kohezije in se zato posledično čez čas razkropi. V drugem primeru, ko je privlačna/odbojna sila dovolj močna, se velikost skupine stabilizira in običajno dobimo natančno določeno povprečno razdaljo med posameznimi osebki. Model obnašanja, ki so ga predstavili Ginelli in sod., je drugačen. Avtorji članka sicer poudarijo, da ta model ni edini, ki lahko predstavi opaženo obnašanje in nepopolno opisuje naravno obnašanje ovc. Je pa uporaben kot dokaz pristopa, saj ponudi vpogled v nekako preprosta individualna pravila interakcije, iz katere lahko izhaja združevanje ovc v čredo. Vse enačbe modela so izpeljane iz pravil, ki so jih zaznali raziskovalci. Stanje vsake ovce je določeno z njeno pozicijo $\mathbf{A}_i^t$, usmerjenostjo premika Θ^t in stanjem obnašanja q_i^t. V modelu so tri stanja obnašanja, ki so mirovanje $\{0\}$, hoja $\{1\}$ ter tek $\{2\}$. Ovca v mirovanju se ne premika in ne spreminja svoje usmerjenosti. Med tem, ko je ovca v stanju hoje ali teka, pa se njena pozicija in usmerjenost spreminjata glede na stanja sosednjih ovc.

3.2.1 Določitev smeri gibanja

Izračun nove pozicije agenta v naslednjem časovnem koraku je podoben, kot pri prejšnjem modelu. Torej, agent se bo v naslednjem časovnem koraku Δt, premaknil iz trenutne pozicije $\mathbf{A}_i^t$ na novo pozicijo $\mathbf{A}_i^{t+\Delta t}$ v smeri $\mathbf{s}_i^{t+\Delta t}$ s hitrostjo $v(q_i^t)$, kot je prikazano v enačbi 3.4. Pri tem modelu je hitrost premika odvisna od stanja obnašanja. Ob enem velja tudi pravilo, da je $v(2) >> v(1) > v(0) = 0$. Smerni vektor premika $\mathbf{s}_i^t$ poračunamo iz usmerjenosti premika Θ_i^t. Ta je predstavljena kot kót med ničelnim vektorjem in smernim vektorjem $\mathbf{s}_i^t$. Smerni vektor $\mathbf{s}_i^t$ dobimo z uporabo enačbe 3.5.

$$\mathbf{A}_i^{t+\Delta t} = \mathbf{A}_i^t + \Delta t v(q_i^t)\mathbf{s}_i^{t+\Delta t} \tag{3.4}$$

$$\mathbf{s}_i^t = (\cos\Theta_i^t, \sin\Theta_i^t) \tag{3.5}$$

Ovca v stanju hoje poskuša poravnati svojo smer z ovcami v M_i. To je, z vsemi sosednjimi ovcami, ki so na razdalji r_0. Smer ovc iz skupine M_i dobimo tako, da ovcam iz te skupine vzamemo usmerjenost premika Θ_i^t ter jih pretvorimo po enačbi 3.5 v vektorje. Iz teh vektorjev nato naredimo vsoto. Ta vsota vektorjev je pravilno usmerjen vektor $\mathbf{V}$, vendar z veliko dolžino. Njegova dolžina nas ne moti, saj ko uporabimo $Arg[\mathbf{V}]$, dobimo kót med vektorjem $\mathbf{V}$ in ničelnim vektorjem, kar je smerni kot $\Theta_i^{t+\Delta t}$. Temu kótu nato še prištejemo šum ψ_i^t, ki je naključno izbran kót iz obsega vrednosti $[-\eta\pi, \eta\pi]$. Celoten postopek je prikazan v enačbi 3.6. Za poenostavitev enačbe avtorji v izračun niso vključili odbijanja med agenti, saj šum poskrbi, da posamezni agenti ne morajo ostati eden na drugem. Zanimivo je, da za razliko od modela obnašanja ovc po Strömbom in sod., pri katerem se ovce gibajo brez poravnave, namreč samo z odboji in privlačnostjo, te pri modelu obnašanja po Ginelli in sod. pri hoji uporabljajo samo poravnavo in ignorirajo odboje ter privlačnost. Zaradi uporabe fizikalnega pogona in komponente za zaznavanja trkov pa do prekrivanja agentov sploh ne more priti, saj ima vsak agent preverjanje trka, ki to prepreči. Z uporabo enačbe 3.6 dosežemo formacijo rahlo usmerjenih lokalnih podskupin ovc, ki se pasejo ter se razkropijo v prostor. S tem ustvarimo podobne vzorce obnašanja, kot so bili opaženi v poskusih.

$$\Theta_i^{t+\Delta t} = Arg[\sum_{j \in M_i} \mathbf{s}_j^t] + \psi_i^t \quad (\text{če je } q_i^t = \{1\}) \tag{3.6}$$

Ovca v stanju teka, ima bolj kompleksni izračun za usmerjenost premika 3.7. Ta se ozira samo na ovce, ki so tudi v stanju teka. Vektor $\hat{\mathbf{e}}_{ij}^t$ je enotni vektor s smerjo od ovce i do sosednje ovce j ter $f(r_{ij}^t)$ je privlačno/odbojna sila, z ravnovesno razdaljo r_e in razdaljo r_{ij}^t, ki meri od ovce i do ovce j. Privlačno/odbojna sila $f(r_{ij}^t)$ je izračunana po enačbi 3.8. To skupaj z enotnim vektorjem $\hat{\mathbf{e}}_{ij}^t$ otežimo z utežjo β ter prištejemo smerni vektor $\mathbf{s}_j^t$ sosednje ovce z utežjo δ. Vsota vseh teh vektorjev je usmerjeni vektor, iz katerega izračunamo smerni kót, kar je končna usmerjenost premika ovce v stanju teka.

$$\Theta_i^{t+\Delta t} = Arg \sum_{j \in V_i} [\delta \mathbf{s}_j^t + \beta f(r_{ij}^t)\hat{\mathbf{e}}_{ij}^t] \quad (\text{če je } q_i^t = \{2\}) \tag{3.7}$$

$$f(r_{ij}^t) = \min(1, \frac{r_{ij}^t - r_e}{r_e}) \tag{3.8}$$

3.2.2 Preklapljanje med stanji obnašanja

Stanje obnašanja q_i^t ovca spreminja glede na stimulacije iz okolja. Opis teh sprememb je definiran z verjetnostmi prehoda med različnimi stanji obnašanja. Raziskovalci so na podlagi opazovanja manjših čred ugotovili, da je verjetnost $p_{0\to1}$ za ovco v mirovanju, da bi začela hoditi, okrepljena s prisotnostjo sosednjih premikajočih se ovc. Za poenostavitev enačbe so avtorji članka sklenili, da se ignorira učinek šibkega zaviranja stacionarnih sosednjih ovc in je verjetnost $p_{1\to0}$, torej verjetnost, da se ovca v stanju hoje ustavi, sestavljena iz iste strukture kot $p_{0\to1}$. Iz tega sledita enačbi 3.9 za prehod v stanje hoje in 3.10 za prehod v stanje mirovanja. V enačbah sta $\tau_{0\to1}$ in $\tau_{1\to0}$ uteži spontanega časovnega prehoda, α utež oponašanja in $n_0^t(i)$ ter $n_1^t(i)$ število sosednjih ovc v mirovanju oz. hoji.

$$p_{0\to1}(i,t) = \frac{1 + \alpha n_1^t(i)}{\tau_{0\to1}} \tag{3.9}$$

$$p_{1\to0}(i,t) = \frac{1 + \alpha n_0^t(i)}{\tau_{1\to0}} \tag{3.10}$$

Prehod v stanje teka ni odvisen od stanja v katerem se nahaja opazovana ovca. Pri prehodu v tek je število ovc v stanju teka $n_2^t(i)$, oteženo z utežjo α, z učinkom skupinskega sočasnega štarta ojačano z eksponentom $\delta > 1$, kjer je l_i^t povprečna razdalja do vseh sosedov, ki so v stanju teka in so na razdalji r_d od ovce i in d_R je karakteristična dolžinska utež za začetek teka. Odnos med tema dvema utežema zagotavlja, da ima na široko razkropljena skupina ovc več možnosti za sprožitev združevanja v čredo, kot pa zgoščena. Enačba 3.11 je tudi otežena z utežjo spontanega časovnega prehoda v tek $\tau_{0,1\to2}$.

$$p_{0,1\to2}(i,t) = \frac{1}{\tau_{0,1\to2}} [\frac{l_i^t}{d_R}(1 + \alpha n_2^t(i))]^\delta \tag{3.11}$$

Za večjo enostavnost modela so se raziskovalci odločili, da je dovolj, če ovca, ki je v stanju teka, lahko prestopi samo v stanje mirovanja. Glej enačbo 3.12, kjer je $\tau_{2\to0}$ utež spontanega časovnega prehoda iz teka mirovanje, l_i^t je povprečna razdalja do vseh sosedov

v zaustavljanju na razdalji r_d ter d_S karakteristična dolžinska utež za konec teka. Pri tem je verjetnost za prestop ojačana s številom sosednjih ovc v zaustavljanju $n^t_{2\to 0}(i)$. To so ovce, ki so v prejšnjem časovnem koraku prestopile iz stanja teka v stanje mirovanja. Druga karakteristična lastnost je, da velja $d_S < d_R$.

$$p_{0,1\to 2}(i,t) = \frac{1}{\tau_{2\to 0}}[\frac{d_R}{l^t_i}(1+\alpha n^t_{2\to 0}(i))]^\delta \tag{3.12}$$

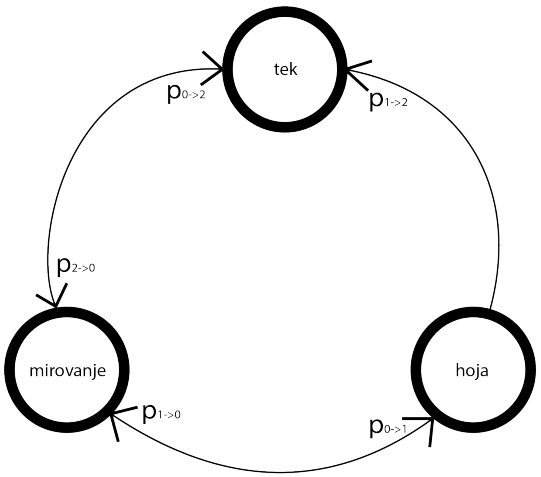
Dobra stran uporabe ustavljajočih se sosedov je, da to privede do nenadne ustavitve celotne črede. Pri prestopu iz teka v mirovanje, ima l^t_i obratno vlogo, kot ga je imel pri prestopu v tek. Vrednost prestopa v mirovanje je ojačana, ko so sosedje v neposredni bližini.

Iz članka ni natančno razvidno, kako naj se uporabi izračunane vrednosti, za ugotavljanje zadostnega pogoja za prestop iz enega stanja v drugo. Zato sem sam predvideval, da so to verjetnosti in sem tako z njimi tudi ravnal. Torej, za stanje v katerem je ovca, izračunam verjetnosti za prestop v drugi stanji oziroma v drugo stanje, če je trenutno v stanju teka. Nato primerjam z naključno vrednostjo, v katero od verjetnosti pade. Če se to ne zgodi, potem ostane v istem stanju kot je sedaj. Prikaz možnih prehodov stanj je prikazan na sliki 3.5, kjer je ob vsaki vhodni puščici tudi pripisana pripadajoča verjetnost prehoda.

3.2.3 Dodano odmikanje od ovčarja

Avtorji članka v svojo raziskavo niso vključili še ovčarja, saj je cilj njihove raziskave bil, da odkrijejo naravno obnašanje ovc brez motenj in groženj. Zato tudi v model obnašanja niso vključili odziv ovce, ko se ji približa ovčar. Ker pa sam v svoji simulaciji delam z ovčarjem, sem modelu obnašanja dodal odmik od grožnje. Primarna smer, v katero se mora ovca odmakniti, da se izogne ovčarju, je izračunana po modelu Strömbom in sod [11] (glej poglavje 3.1.3). Izračunani smerni vektor nato normaliziram ter ga prištejem smeri gibanja, ki je izračunana glede na sosednje ovce. To storim, zato da imata oba vektorja enako težo pri določanju končne smeri premika.

V naravi ovce začnejo teči, ko se jim približa grožnja. Prav zato agent, ko se mu ovčar dovolj približa, da ga zazna, preklopi v stanje teka. To je tudi vzrok, da se smer gibanja



Slika 3.5 Diagram prehajanja stanj.

glede na ovčarja popravlja samo, ko je ovca v stanju teka.

Večino vrednosti parametrov sem uporabil takih, kot so navedene v članku. Nekaj pa sem jih popravil, ter tako izboljšal delovanje. Zmanjšal sem vrednost d_S, kar je pripomoglo k manjši verjetnosti za prestop iz teka v mirovanje. Ovce so zaradi tega ostale v teku dlje časa in se združile v čredo, kakor je opisano v članku. Povečal sem tudi obe vrednosti za spontani časovni prehod iz mirovanja v hojo in obratno, ter s tem zmanjšal izračunano verjetnost za prehod med stanji, kar mi je omogočilo boljše delovanje, saj so verjetnosti prej velikokrat prišle nad vrednost 1. Vse vrednosti, ki sem jih uporabil pri izdelavi modela obnašanja po Ginelli in sod., so navede v tabeli 3.2.

Tabela 3.2 Seznam uporabljenih parametrov v simulaciji z veličinami za model obnašanja po Ginelli in sod. [12].

Parameter	Opis	Uporabljena vrednost
v_1	Hitrost premikanja v stanju hoje	2.5
v_2	Hitrost premikanja v stanju teka	5
$\tau_{0\to1}$	Spontani časovni prehod iz mirovanja v hojo	70
$\tau_{1\to0}$	Spontani časovni prehod iz hoje v mirovanje	16
$\tau_{0,1\to2}$	Spontani časovni prehod v tek	N - št. ovc v simulaciji
$\tau_{2\to0}$	Spontani časovni prehod iz teka v mirovanje	N - št. ovc v simulaciji
d_R	Karakteristična dolžinska utež za prestop v tek	31.6
d_S	Karakteristična dolžinska utež za prestop iz teka	2.1
r_e	Ravnovesna razdalja za izračun privlačno/odbojne sile	1
r_0	Interakcijska razdalja med hojo	1
r_d	Območje zaznavanja sosednjih agentov	15
r_s	Območje zaznavanja ovčarja	30
α	Utež efekta oponašanja	15
β	Relativna moč privlačno/odbojne sile	0.8
δ	Ojačevalni eksponent	4
η	Razpon šuma	0.13

3.3 Model obnašanja ovčarja

Ovčarjeva naloga je, da zbere vse ovce v eno čredo in jo pripelje v manjši pašnik. Njegova naloga je zaključena, ko zadnja ovca vstopi na ciljno področje. V članku avtorjev Strömbom in sod. [11], je poleg modela obnašanja ovc opisan tudi preprost model obnašanja ovčarja. Ta ima, v času vodenja ovc, na voljo dve možnosti: zbiranje ali usmerjanje ovc. Za katero od možnosti se bo odločil je odvisno od tega, ali so vsi agenti v razdalji $f(N)$ od GCM. GCM nam predstavlja center črede opazovanih ovc. Za izračun lokacije GCM, preprosto vzamemo vsoto pozicij vseh agentov in jo delimo z njihovim številom, kot prikazuje enačba 3.13. To lahko storimo zato, ker privzamemo, da so vse ovce enake. Za določitev dovoljene velikosti črede $f(N)$, uporabimo število ovc N, ki jih imamo v simulaciji. Pri tem je $f(N)$ dovolj velika razdalja, da dopušča tudi nepravilno obliko črede in je podana z enačbo 3.14. Ko korenimo število agentov N, dobimo območje v katerem bi lahko bile vse ovce. Ker pa čreda ovc nikoli ni popolno okrogla in imajo ovce tudi med seboj nekaj prostora, radij kroga povečamo za r_a, ki hkrati predstavlja razdaljo na kateri se začnejo ovce odmikati ena od druge.

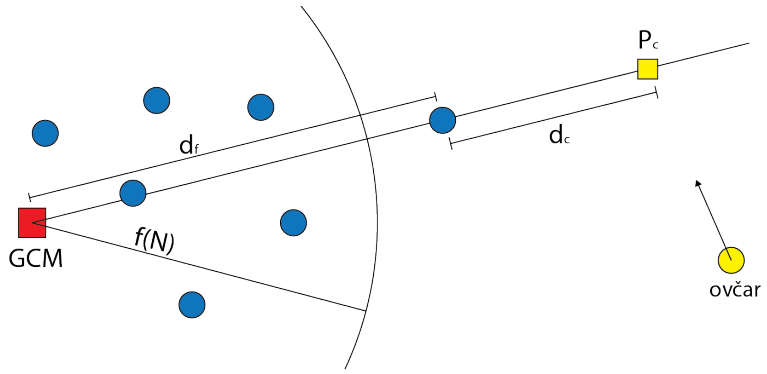
$$\mathbf{GCM} = \frac{\sum_{j=1}^{N} \mathbf{A}_j}{N} \tag{3.13}$$

$$f(N) = r_a\sqrt{N} \tag{3.14}$$

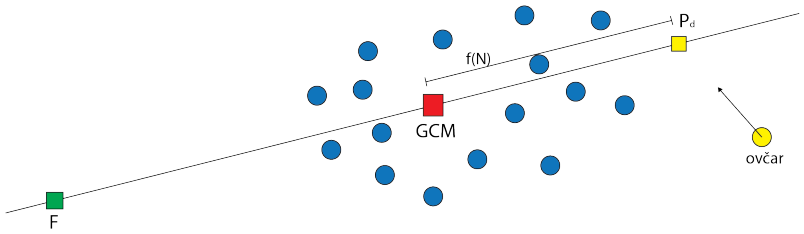
3.3.1 Zbiralno stanje ovčarja

V primeru, da je kakšna ovca zunaj območja $f(N)$, ovčar začne z zbiranjem ovc. To stori tako, da se ovčar pomakne na pozicijo $\mathbf{P}_c$, da prežene najbolj odmaknjene agente v čredo. Pozicija $\mathbf{P}_c$ je lokacija, na katero se mora postaviti ovčar, da se ovca odmika od ovčarja v smeri proti GCM, kot prikazuje slika 3.6.

Točko $\mathbf{P}_c$ izračunamo tako, da je, če je $\mathbf{A}_f$ pozicija najbolj oddaljenega agenta na razdalji d_f od GCM in je d_c razdalja na kateri ovčar vodi ovco v čredo, izračun pozicije definiran po enačbi 3.15. Bolj preprosto, poziciji GCM odštejemo pozicijo najbolj oddaljenega agenta $\mathbf{A}_f$ in to normaliziramo. S tem dobimo smerni vektor, ki kaže od globalnega težišča proti ovci. Ta vektor nato pomnožimo z razdaljo d_c, ker mu s tem določimo dolžino, in ga prištejemo poziciji najbolj oddaljene ovce. Rezultat je točka, ki se nahaja



Slika 3.6 Zbiranje ovc v čredo.



Slika 3.7 Vodenje ovc do cilja.

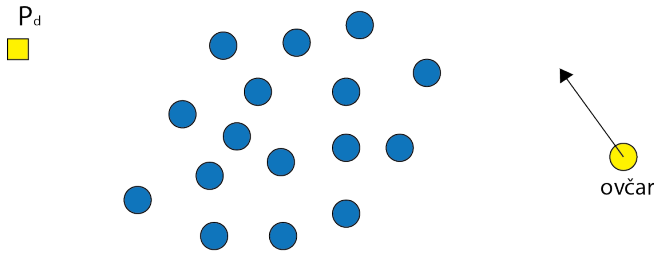
za agentom na razdalji d_c.

$$\mathbf{P}_c = \mathbf{A}_f + \frac{\mathbf{GCM} - \mathbf{A}_f}{||\mathbf{GCM} - \mathbf{A}_f||} d_c \tag{3.15}$$

3.3.2 Gonilno stanje ovčarja

Usmerjanje ovc do cilja se zgodi, ko so vse ovce znotraj območja $f(N)$ od GCM. V tem primeru se ovčar premakne na pozicijo $\mathbf{P}_d$, da usmeri čredo proti cilju. Točka $\mathbf{P}_d$ se nahaja za čredo proti cilju, kot prikazuje slika 3.7.

Izračun točke $\mathbf{P}_d$ je definiran po enačbi 3.16, pri čemer je $\mathbf{F}$ ciljna lokacija in $f(N)$ dovoljena velikost črede. Izračunana vrednost je pozicija točke, ki se nahaja na repu črede usmerjene proti cilju. Enačba je zelo podobna kot prejšnja, saj se mora ovčar, pri vodenju do željene lokacije, nahajati za posameznikom ali skupino. Da dobimo smerni



Slika 3.8 Primer obvoza ovčarja za izognitev negativnega grupiranja.

vektor poganjanja, odštejemo ciljno točko **F** od GCM ter dobljeni vektor normaliziramo. Ker je pri prejšnji enačbi ovčar vodil samo eno ovco, se je kot osnova za izračun točke vzela lokacija tega posameznika. Sedaj, ko mora ovčar voditi skupino, pa vzamemo za osnovo GCM, ki nam predstavlja center črede. Torej GCM prištejemo normalizirani smerni vektor poganjanja, ki ga pomnožimo z dovoljeno velikostjo črede $f(N)$.

$$\mathbf{P}_d = \mathbf{GCM} + \frac{\mathbf{GCM} - \mathbf{F}}{||\mathbf{GCM} - \mathbf{F}||} f(N) \tag{3.16}$$

3.3.3 Izboljšave ovčarja

Končno določanje smeri, v kateri se bo gibal ovčar v naslednjem časovnem koraku, sem izboljšal z dodanim izogibanjem ovcam na poti. S tem sem želel zmanjšati možnosti, da pride do negativnega grupiranja. To je dogodek, ko ovčar začne početi ravno nasprotno kot je njegova naloga. Primer tega je, da se čreda znajde med ovčarjem in njegovo destinacijo. Ko se začne ovčar premikati proti svojemu cilju, se mu ovce umikajo in pride do razkropitve črede ali pa se mu cela čreda skupaj umika, kar povzroči mrtvo situacijo, saj se tako ovčarju premika tudi njegov cilj. En del rešitve je, da se pri izračunu cilja za ovčarja upošteva šum. Dodal pa sem tudi, da če ima ovčar na svoji poti čredo oziroma posamezno ovco in je ta na razdalji d_a, potem bo ovčar spremenil svojo pot za kot 45°. S tem bo šel okoli agentov in ne med njih, kot prikazuje slika 3.8.

V članku Strömbom in sod. [11] je opisan ovčar, ki ima na voljo dve hitrosti: se giblje ali stoji. Ko se giblje, se giblje s hitrostjo v_s in je njegovo gibanje hitrejše od ovc. V primeru,

da se preveč približa ovcam pa obstane na mestu ter jim da čas, da se mu umaknejo. Za namen preproste simulacije je to dovolj. Ker pa sem v svoji implementaciji uporabil tudi model obnašanja ovc po Ginelli in sod. [12], ki uporablja za ovce tri hitrosti gibanja, sem dodal ovčarju še eno hitrost. Ta je vmesna hitrost in je enako velika, kot jo imajo ovce v stanju teka. S tem se ovčarju ni več potrebno občasno ustavljati, ker bi prišel preblizu, temveč lahko potuje skupaj z ovcami. S srednjo hitrostjo se začne premikati, ko je razdalja do ovc manjša od d_0 ali pa je cilj v razdalji d. Upočasnitev blizu cilja je z namenom, da se prepreči preskok cilja.

V zbiralnem stanju mora ovčar prehiteti ovco, ki mu je ušla, in jo usmeriti nazaj v čredo. V tem primeru ovčar zato pohitri premikanje. Zelo malo verjetno je, da je še kakšna ovca blizu poti, ki jo mora opraviti ovčar, da pride do pobegle ovce. Tudi če bi bila in bi moral ovčar zaradi zaznane bližine upočasniti, bi se mu ta hitro umaknila in bi lahko nadaljeval s polno hitrostjo.

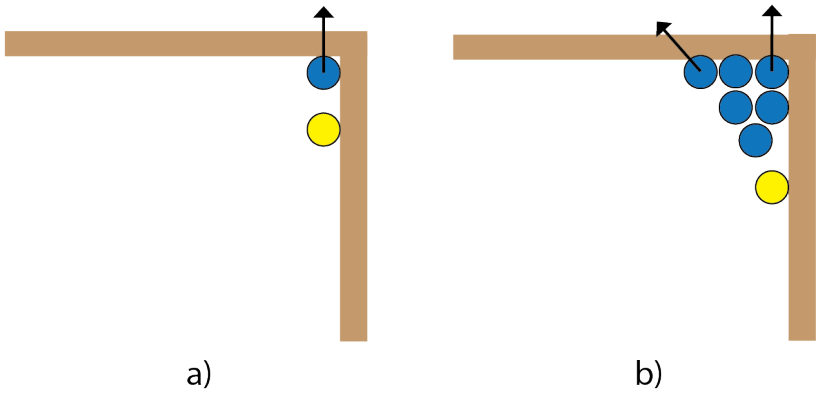
Vendar pa še vedno lahko pride to tega, da se jim preveč približa. To se lahko na primer zgodi, ko ovce pridejo do ograje in ne morejo napredovati v željeni smeri. Takrat mora cela čreda obrniti svojo smer in to lahko tudi nekaj časa traja. Ovčar pa se zato, ker še ni pri ograji in ga ta ne ovira, še kar približuje čredi. Šele nato zazna, da je nekaj narobe in jim da čas, da popravijo svojo smer. Parametri, ki sem jih uporabil v simulaciji, za model obnašanja psa so navedeni v tabeli 3.3.

Tabela 3.3 Seznam uporabljenih parametrov v simulaciji z njihovimi vrednostmi za model obnašanja ovčarja.

Parameter	Opis	Uporabljena vrednost
v_1	Hitrost premikanja v stanju vodenja ovc	5
v_2	Hitrost premikanja v stanju iskanja ovc	7.5
r_a	Faktor za dovoljeno velikost črede	2
d_c	Razdalja za zbiranje ovc v čredo	10
d_a	Razdalja za zaznavo ovc na poti	20
d_0	Razdalja za upočasnitve v bližini ovc	10
d_f	Razdalja za upočasnitev v bližini cilja	5
e	Relativna moč šuma	0.3

4 Testiranje in analiza rezultatov

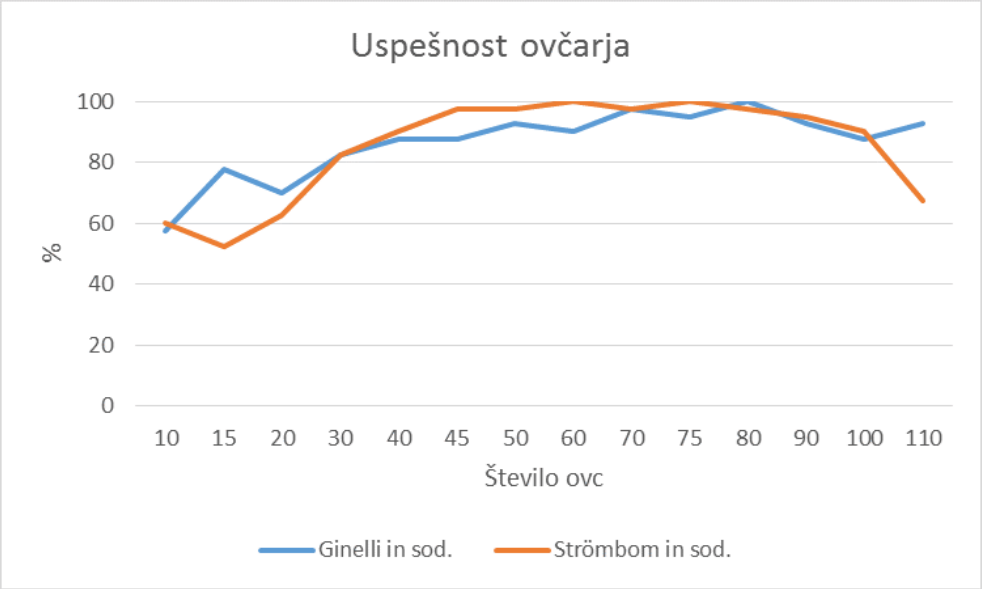
Po izdelavi vseh treh modelov obnašanja, se lahko lotimo testiranja uspešnosti ovčarja pri obeh modelih ovc. Tega se lotimo tako, da nastavimo začetno pozicijo ovčarja v ciljno ograd o ter postavimo potrebno število ovc za simulacijo na naključna mesta na pašniku. Število ovc na pašniku spreminjamo zato, da opazujemo, kako se ovčar odziva pri različnem številu ovc. Pri vsakem testiranju simulacije tudi določimo, kateri model obnašanja ovc bomo uporabili pri simulaciji. Tako ima vsako testiranje dve spremenljivki, od katerih je odvisen končni rezultat. Da pridemo do bolj točnih rezultatov testiranja, vsako simulacijo ponovimo večkrat. S tem dobimo boljši vzorec, iz katerega lahko pridobimo podatke o času, ki ga potrebuje ovčar, da konča svojo nalogo. Pri testiranju sem uporabil za vsako simulacijo 40 ponovitev. Omejil sem tudi čas, ki ga lahko porabi ovčar, za uspešno opravljeno nalogo. Določil sem, da lahko ovčar porabi največ 200 sekund, da opravi svojo nalogo. Če mu ne uspe končati v tem času, se simulacija šteje za neuspešno. Do časovne omejitve sem prišel s predhodnimi testiranji, pri katerih sem nekajkrat pognal simulacijo z različnimi števili ovc in opazoval čas, ki je bil potreben za uspešen zaključek. Pri teh testiranjih je bila časovna zahtevnost vedno veliko manjša,



Slika 4.1 a) Primer mrtve situacije in b) umik črede stisnjene v kot.

zato sem se odločil, da je ta vrednost dovolj velika za zaznavo mrtve situacije. Do tega dogodka pride v primerih, ko se posamezna ovca postavi v kót polja. V tem primeru je ovčar nemočen, saj ovce ne more spraviti iz kota. To se zgodi zato, ker se želi ovca, ko se ji ovčar približa, umakniti, vendar pa ne more nikamor, saj je stisnjena v kót. Pri tem tudi ni pomembno iz katere smeri se ji ovčar približa, saj ima vedno pred sabo oviro, kot je prikazano na sliki 4.1 a). Drugače je, če imamo primer črede, ki je stisnjena v kót, saj se jim v tem primeru lahko ovčar približa iz strani in se ovce, ki so na drugi strani črede, začnejo umikati poševno in potegnejo s seboj celotno čredo, kot je prikazano na sliki 4.1 b). Seveda se ovca, ki je najbolj v kótu, želi umakniti od ovčarja naravnost v ograjo. Vendar pa, ko se začnejo ostale ovce odmikati stran, jo čredni nagon potegne z njimi.

Da do tega ne bi prihajalo, bi bilo potrebno nadgraditi oba modela obnašanja ovc. Vključiti bi bilo potrebno zaznavo ograje in izogib trka z njo. Ob tem bi lahko nastalo nekaj težav, saj je potrebno vključiti tudi odločitev, v katero smer se bo ovca umaknila. V primeru, ko bi se ovca premikala naravnost proti ograji, bi jo lahko preprosto preusmerili na levo ali desno, mogoče celo glede na lokacijo najbližje črede. Če bi se ovca približevala ograji poševno, potem ima na možnost umik od ograje samo na eno stran. Večji problem je umik iz kóta, ko je v bližini ovčar. V tem primeru, bi se morala ovca, da bi se rešila, umakniti proti grožnji. To pa je ovcam in tudi ostalim živalim proti nagonu, kar pomeni



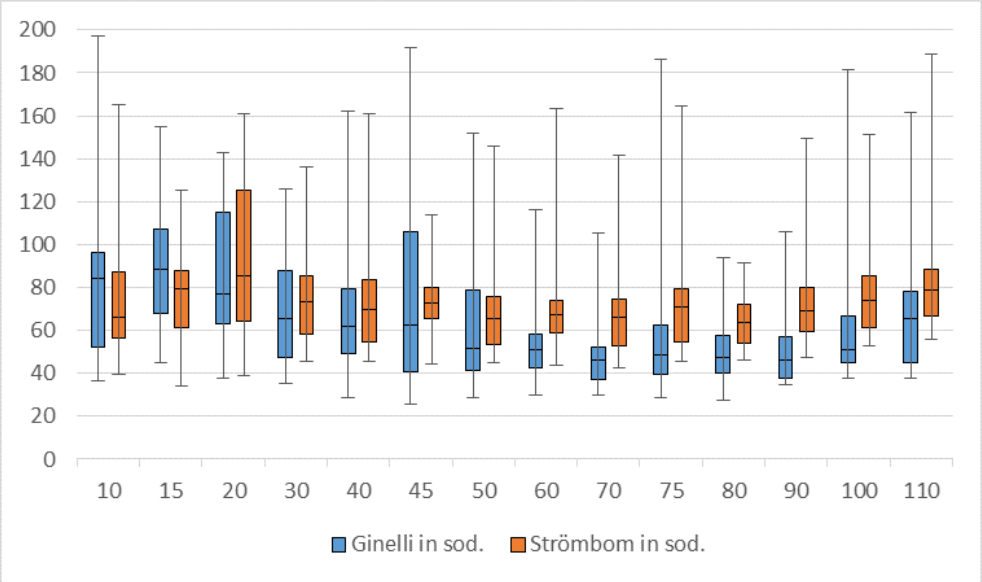
Slika 4.2 Graf uspešnosti ovčarja glede na število ovc in izbrani model obnašanja ovc.

da je ujeta v past iz katere se ne more rešiti. Take primere, ko se posamezna ovca ali pa manjša skupina ovc ujame v kót ograde, najdemo tudi v resničnem svetu.

Za testiranje sem nastavil velikost intervala števila ovc do 110. Za to število sem se odločil, ker je dovolj veliko, da predstavim ogromno čredo ovc, ob enem pa simulacija nima težav s prezahtevnim računanjem. Na tem intervalu sem najprej pognal simulacije s korakom 10. To pomeni, da je število ovc, ki so bile na polju, večkratnik števila 10. Za večjo natančnost končnih rezultatov, sem pognal še nekaj vmesnih simulacij. Oba modela obnašanja sem simuliral z istim številom ovc s 40-imi ponovitvami. Podatke, ki sem jih pridobil iz posamezne simulacije[1] sem analiziral in pridobil rezultate, ki jih prikazujejo grafi za boljšo preglednost.

Uspešnost ovčarja U_s pri določenem številu ovc sem poračunal tako, da sem število uspešno končanih simulacij k_s delil s številom vseh opravljenih simulacij, kot prikazuje enačba: $U_s = k_s/40$. Uspešnost ovčarja se pri obeh modelih obnašanja povečuje, ko se povečuje število ovc na polju. Vendar pa se pri velikem številu ovc spet nekoliko zmanjša,

[1] Video primera simulacije: https://youtu.be/HuB21v92YC0



Slika 4.3 Graf časovne zahtevnosti glede na število ovc in izbrani model obnašanja ovc.

kot prikazuje slika 4.2. Do manjše uspešnosti pri majhnem številu ovc pride, ker so ovce na široko razkropljene po polju in se težje zaznajo med sabo. Posledica tega je, da se težje združijo v čredo in je zato naloga ovčarja, da jih pripelje dovolj skupaj, da so v območju medsebojnega zaznavanja. Ker je pogosteje, da je ovca sama in jo mora ovčar iti iskat, se hitreje zgodi, da pride do dogodka, ko se ovca ujame v mrtvo situacijo. Pri povečevanju števila ovc na polju, se prostor med posamezniki zmanjšuje in se lažje in hitreje združijo v čredo, kar pripomore k izogibu mrtve situacije. Ko število ovc preveč naraste, pa zaradi pomanjkanja prostora na polju hitreje pride do negativnega združevanja, ko se ovčar poskuša postaviti na svoje mesto. Posledica je, da ovčar nehote potisne ovco v kót in je nato ne more več rešiti iz njega. Ko opazujemo graf, opazimo da se uspešnost ovčarja pri manjšem številu ovc hitreje dviguje pri modelu obnašanja po Ginelli in sod. [12], kot pa pri modelu obnašanja po Strömbom in sod. [11]. Vendar pa, ko prekoračimo število 30, se situacija obrne. Pri velikem številu ovc na polju, pa je ovčar spet bolj uspešen pri modelu po Ginelli in sod. Iz česar sledi, da je ovčar v robnih primerih bolj uspešen pri modelu po Ginelli in sod. ter pri vmesnih primerih nekoliko uspešnejši pri modelu po Strömbom in sod.

Graf na sliki 4.3 je sestavljen iz pokončnih črt in pravokotnikov na teh črtah. Črta

je razpon zabeleženega potrebnega časa do uspeha ovčarja. Torej, najnižja točka črte predstavljaj najkrajši zabeležen čas ter obratno, najvišja točka črte predstavlja najdaljši zabeležen čas. Pravokotnik predstavlja razpon časa, v katerem je ovčar najbolj pogosto uspešno končal nalogo. Vodoravna črta na sredini pravokotnika označuje povprečni čas, ki ga je potreboval ovčar, do uspešno opravljene simulacije. Višina pravokotnika pa je enaka dvojnemu standardnemu odklonu. Standardni odklon nam predstavlja, kako razpršene so vrednosti okoli aritmetične sredine. To pomeni, večji kot je standardni odklon, bolj so vrednosti razpršene in obratno, bolj kot je majhen, bolj so vrednosti zgoščene. Oba modela sta na grafu predstavljena s svojo barvo in sicer model po Ginelli in sod. je predstavljen z modro barvo ter model po Strömbom in sod. je predstavljen z oranžno barvo.

Najkrajši zabeležen čas do uspeha ovčarja se pri modelu obnašanja po Strömbom in sod. [11] dviguje s povečevanjem števila ovc, kot se vidi na grafu slike 4.3, če opazujemo spodnje točke pokončnih črt oranžnih pravokotnikov. Medtem pa se pri modelu obnašanja po Ginelli in sod. [12], najkrajši čas spušča s povečevanjem števila ovc in se, nato spet dvigne, kot lahko opazimo na grafu slike 4.3, ko opazujemo spodnje točke pokončnih črt modrih pravokotnikov. Pri obeh modelih najdaljši čas do uspeha ovčarja zelo niha, kot prikazuje graf na sliki 4.3, ko opazujemo najvišje točke pokončnih črt. Vendar pa ima model po Ginelli in sod. večji nihajni razpon. Pri opazovanju najkrajšega in najdaljšega zabeleženega potrebnega časa, se moramo zavedati, da so to ekstremni podatki in da iz njih ne moremo zares sklepati, kateri model je boljši oziroma pri katerem modelu je ovčar bolj uspešen.

Pri opazovanju grafa na sliki 4.3 vidimo, da se pri modelu obnašanja po Strömbom in sod. povprečni čas do uspeha ovčarja ne spremeni veliko s povečevanjem števila ovc. Pri majhnem številu ovc je povprečni čas blizu 80 sekund, ko pa povečujemo število ovc se spusti proti 70 sekund ter se nato dvigne nazaj do 80. Podobno je s povprečnim časom pri modelu obnašanja po Ginelli in sod., kjer je pri majhnem številu ovc vrednost nad 80 sekund in se nato, s povečevanjem števila ovc, postopoma zmanjšuje proti 50 sekund. Pri velikem številu ovc, tudi pri tem modelu povprečni čas nazaj raste. Opazimo tudi, da je na grafu slike 4.3 standardni odklon pri modelu obnašanja po Ginelli in sod. v povprečju vedno nekoliko večji, kot pa pri modelu obnašanja po Strömbom in sod. Kar

pomeni, da se pri modelu po Ginelli in sod. izmerjeni časi dosti bolj razlikujejo.

5 Sklepne ugotovitve

V diplomski nalogi sem poustvaril modele obnašanja ovc in psa ovčarja, ki so predstavljeni v člankih [11, 12]. Dodal sem tudi nekaj izboljšav, s katerimi sem dosegel večjo uspešnost simulacije. Nato sem opravil testiranje in pridobljene rezultate analiziral ter jih ovrednotil. Glede na rezultate, ki sem jih dobil pri testiranju, lahko sklepam, da je ovčar bolj uspešen pri modelu obnašanja po Ginelli in sod. [12], čeprav uspešnost pri tem modelu ne doseže tako hitro vrha uspešnosti, kot pri modelu obnašanja po Strömbom in sod. [11]. Do tega sklepa sem prišel zaradi tega, ker ovčar v povprečju hitreje konča svojo nalogo, čeprav se časi med seboj veliko bolj razlikujejo.

Pri testiranju sem opazil, da je velikokrat prišlo da združevanja ovc v manjše črede. Te črede so se včasih nahajale tudi blizu ciljnega polja. Ker pa ima ovčar v svojem algoritmu določeno, da mora združiti vse ovce v eno čredo, ki jo pripelje na cilj, je moral te manjše skupine odpeljati stran od cilja in v večjo čredo. Tukaj bi lahko dodali izboljšavo, da ovčar manjše črede ovc posamezno vodi do ciljne ravnine. S tem ne bi več izgubljali časa, ko mora ovčar odpeljati ovce stran od cilja in nato spet nazaj, ter tako dosegli veliko

pohitritev simulacije. Ker ovčarski psi lahko sami čredijo več kot sto ovc hkrati, bi ta izboljšava služila bolj za boljše rezultate pri simulaciji, kot pa poustvarjanje naravnega vedenja. Zanimiva nadgradnja simulacije obnašanja bi bila vključitev več ovčarskih psov. V tem primeru bi psi lahko hitreje zbrali ovce v čredo, saj bi se vsak približeval iz svoje strani in bi tako potisnili ovce skupaj. Tudi vodenje ovc bi bilo lažje, saj bi le en ali dva psa opravljala nalogo popravljanja črede, to je lovljenje ovc, ki se oddaljijo od črede, medtem ko bi ostali potiskali čredo proti cilju. Naslednja zanimiva nadgradnja bi bila vključitev volkov v simulacijo. Tukaj bi morali ovčarji, poleg vodenja ovc, opravljati tudi nalogo branjenja ovc pred napadi. Možnih je veliko variacij simulacije, kjer napade posameznik ali pa različno velik trop volkov. Doda se tudi izguba ovce ali pa ovčarja, če se spopade s tropom volkov. Za ugotavljanje uspešnosti simulacije bi lahko ugotavljali koliko ovc je bilo uspešno pripeljanih oziroma koliko jih je umrlo, koliko ovčarjev je izgubilo življenje, koliko napadov je bilo uspešno odbitih in koliko ne ter ali je uspelo ovčarjem pokončati katerega od volkov. Tukaj bi lahko tudi dodali, da ima posameznik tri stopnje življenja: nepoškodovan, ranjen, mrtev.

Literatura

[1] C. Vo, Robust and reusable methods for shepherding and visibility-based pursuit, PhD dissertation, George Mason University (2014).

[2] J.-M. Lien, S. Rodriguez, J.-P. Malric, N. M. Amato, Shepherding behaviors with multiple shepherds, in: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, IEEE, 2005, pp. 3402–3407.

[3] C. W. Reynolds, Flocks, herds and schools: A distributed behavioral model, ACM SIGGRAPH computer graphics 21 (4) (1987) 25–34.

[4] L. E. Parker, B. Kannan, X. Fu, Y. Tang, Heterogeneous mobile sensor net deployment using robot herding and line-of-sight formations, in: Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, Vol. 3, IEEE, 2003, pp. 2488–2493.

[5] M. K. Kang, J. S. Lee, Application of herding problem to a mobile robot, Journal of Institute of Control, Robotics and Systems 11 (4) (2005) 322–329.

[6] A. Davids, Urban search and rescue robots: from tragedy to technology, IEEE Intelligent Systems 17 (2) (2002) 81–83.

[7] N. M. Kakalis, Y. Ventikos, Robotic swarm concept for efficient oil spill confrontation, Journal of hazardous materials 154 (1) (2008) 880–887.

[8] S. Gade, A. A. Paranjape, S.-J. Chung, Herding a flock of birds approaching an airport using an unmanned aerial vehicle, in: AIAA Guidance, Navigation, and Control Conference. Kissimmee, FL: AIAA, 2015.

[9] M. Trafankowski, Comparing algorithms for simulation of flocking and herding behaviour, Tech. rep. (2010).

[10] B. Bennett, M. Trafankowski, A comparative investigation of herding algorithms, in: Proc. Symp. on Understanding and Modelling Collective Phenomena (UMoCoP), 2012, pp. 33–38.

[11] D. Strömbom, R. P. Mann, A. M. Wilson, S. Hailes, A. J. Morton, D. J. T. Sumpter, A. J. King, Solving the shepherding problem: heuristics for herding autonomous, interacting agents, Journal of The Royal Society Interface 11 (100).

[12] F. Ginelli, F. Peruani, M.-H. Pillot, H. Chaté, G. Theraulaz, R. Bon, Intermittent collective dynamics emerge from conflicting imperatives in sheep herds, Proceedings of the National Academy of Sciences 112 (41) (2015) 12729–12734.

[13] G. Kramer, Experiments on bird orientation, Ibis 94 (2) (1952) 265–285.