

MIN Faculty Department of Informatics



Mobile robot avoid obstascles using Monocular Vision

Hongzhuo Liang



University of Hamburg Faculty of Mathematics, Informatics and Natural Sciences Department of Informatics

Technical Aspects of Multimodal Systems

28. Nov. 2017



Outline

1. Introduction

Motivation Related Work

2. Details

Try to use CNN Use self supervised learning Use Reinforcement Learning

3. Experiments

Experiment 1 Experiment 2

- 4. Conclusion
- 5. Future Work
- 6. Bibliography



Outline

| Introduct | ion Details | Experiments | Conclusion | Future Work | Bibliography |
|-----------|--|-------------|------------|-------------|--------------|
| 1. lr | t <mark>roduction</mark> Motivation Related Work | | | | |
| 2. D | etails | | | | |
| 3. E | xperiments | | | | |
| 4. C | onclusion | | | | RO |
| 5. F | uture Work | | | | ananan |
| 6. B | ibliography | | | | NG TO DA |
| | | | | | |
| | | | | | 夏川夏云烈 |
| | | | | | |
| | | | | | |
| | | | | | 豊葉豊美 |
| | | | | | |
| | | | | | |

Motivation: Why use Monocular Vision

Bil

Bibliography

In the paper *Deep Learning* from Nature: [LeCun et al., 2015]¹

The future of deep learning: Unsupervised learning had a catalytic effect in reviving interest in deep learning, we expect unsupervised learning to become far more important in the longer term. Human and animal learning is largely unsupervised: we discover the structure of the world by observing it, not by being told the name of every object.

¹https://www.nature.com/articles/nature14539

Motivation: Why use Monocular Vision

In the paper *Deep Learning* from Nature: [LeCun et al., 2015]¹

The future of deep learning: Unsupervised learning had a catalytic effect in reviving interest in deep learning, we expect unsupervised learning to become far more important in the longer term. Human and animal learning is largely unsupervised: we discover the structure of the world by observing it, not by being told the name of every object.

¹https://www.nature.com/articles/nature14539



| Introduction | Details | Experiments | Conclusion | Future Work | Bibliography |
|--------------|---------|-------------|------------|-------------|--------------|
| | | | | | |

Advantage

- Cheap
- Easy to get
- Flexible to complicated environments

- Not as accuracy as depth sensor
- Need a lot of calculate power
- Need a lot of training
- Not stable in current time





| Introduction | Details | Experiments | Conclusion | Future Work | Bibliography |
|--------------|---------|-------------|------------|-------------|--------------|
| | | | | | |

Advantage

- Cheap
- Easy to get
- Flexible to complicated environments

- Not as accuracy as depth sensor
- Need a lot of calculate power
- Need a lot of training
- Not stable in current time





| Introduction | Details | Experiments | Conclusion | Future Work | Bibliography |
|--------------|---------|-------------|------------|-------------|--------------|
| | | | | | |

Advantage

- Cheap
- Easy to get
- Flexible to complicated environments

- Not as accuracy as depth sensor
- Need a lot of calculate power
- Need a lot of training
- Not stable in current time





| Introduction | Details | Experiments | Conclusion | Future Work | Bibliography |
|--------------|---------|-------------|------------|-------------|--------------|
| | | | | | |

Advantage

- Cheap
- Easy to get
- Flexible to complicated environments

- Not as accuracy as depth sensor
- Need a lot of calculate power
- Need a lot of training
- Not stable in current time





| Introduction | Details | Experiments | Conclusion | Future Work | Bibliography |
|--------------|---------|-------------|------------|-------------|--------------|
| | | | | | |

Advantage

- Cheap
- Easy to get
- Flexible to complicated environments

- Not as accuracy as depth sensor
- Need a lot of calculate power
- Need a lot of training
- Not stable in current time





| Introduction | Details | Experiments | Conclusion | Future Work | Bibliography |
|--------------|---------|-------------|------------|-------------|--------------|
| | | | | | |

Advantage

- Cheap
- Easy to get
- Flexible to complicated environments

- Not as accuracy as depth sensor
- Need a lot of calculate power
- Need a lot of training
- Not stable in current time





| Introduction | Details | Experiments | Conclusion | Future Work | Bibliography |
|--------------|---------|-------------|------------|-------------|--------------|
| | | | | | |

Advantage

- Cheap
- Easy to get
- Flexible to complicated environments

- Not as accuracy as depth sensor
- Need a lot of calculate power
- Need a lot of training
- Not stable in current time





| Introduction | Details | Experiments | Conclusion | Future Work | Bibliography |
|--------------|---------|-------------|------------|-------------|--------------|
| | | | | | |

Advantage

- Cheap
- Easy to get
- Flexible to complicated environments

- Not as accuracy as depth sensor
- Need a lot of calculate power
- Need a lot of training
- Not stable in current time



| Introduction | | | |
|--------------|--|--|--|
| | | | |

Advantage

- Cheap
- Easy to get
- Flexible to complicated environments

- Not as accuracy as depth sensor
- Need a lot of calculate power
- Need a lot of training
- Not stable in current time



Introduction

periments

Conclusio

Future V

Bibliography

In the traditional way of doing obstacle avoidance task, one should use the camera to detect the point flow to get the information about the environment. For example, the position of the floor and obstacles. Then we can use some fixed way to plan and executes.

- Use Depth camera to get Point Cloud
- Use the Point Cloud to fitting a function about the ground
- Calculate the distance between every point with the fitting ground h
- If h is larger than a setting threshold, then marke it as an obstacle
- Transfer the Point Cloud data of the obstacle to the robot coordinate space
- Combine the current robot moving stats with the obstacle to decide how to avoid the obstacles

šš:_____

xperiments

Conclusio

Future \

Bibliography

In the traditional way of doing obstacle avoidance task, one should use the camera to detect the point flow to get the information about the environment. For example, the position of the floor and obstacles. Then we can use some fixed way to plan and executes.

- Use Depth camera to get Point Cloud
- Use the Point Cloud to fitting a function about the ground
- Calculate the distance between every point with the fitting ground h
- If h is larger than a setting threshold, then marke it as an obstacle
- Transfer the Point Cloud data of the obstacle to the robot coordinate space
- Combine the current robot moving stats with the obstacle to decide how to avoid the obstacles

Introduction

Conclusion

Future W

Bibliography

In the traditional way of doing obstacle avoidance task, one should use the camera to detect the point flow to get the information about the environment. For example, the position of the floor and obstacles. Then we can use some fixed way to plan and executes.

- Use Depth camera to get Point Cloud
- Use the Point Cloud to fitting a function about the ground
- Calculate the distance between every point with the fitting ground h
- If h is larger than a setting threshold, then marke it as an obstacle
- Transfer the Point Cloud data of the obstacle to the robot coordinate space
- Combine the current robot moving stats with the obstacle to decide how to avoid the obstacles

In the traditional way of doing obstacle avoidance task, one should use the camera to detect the point flow to get the information about the environment. For example, the position of the floor and obstacles. Then we can use some fixed way to plan and executes.

- Use Depth camera to get Point Cloud
- Use the Point Cloud to fitting a function about the ground
- Calculate the distance between every point with the fitting ground h
- If h is larger than a setting threshold, then marke it as an obstacle
- Transfer the Point Cloud data of the obstacle to the robot coordinate space
- Combine the current robot moving stats with the obstacle to decide how to avoid the obstacles

In the traditional way of doing obstacle avoidance task, one should use the camera to detect the point flow to get the information about the environment. For example, the position of the floor and obstacles. Then we can use some fixed way to plan and executes.

- Use Depth camera to get Point Cloud
- Use the Point Cloud to fitting a function about the ground
- Calculate the distance between every point with the fitting ground h
- If h is larger than a setting threshold, then marke it as an obstacle
- Transfer the Point Cloud data of the obstacle to the robot coordinate space
- Combine the current robot moving stats with the obstacle to decide how to avoid the obstacles

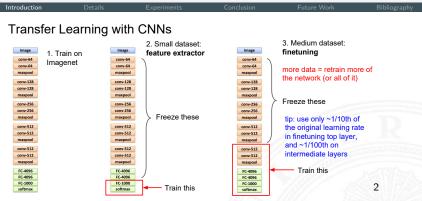
In the traditional way of doing obstacle avoidance task, one should use the camera to detect the point flow to get the information about the environment. For example, the position of the floor and obstacles. Then we can use some fixed way to plan and executes.

- Use Depth camera to get Point Cloud
- Use the Point Cloud to fitting a function about the ground
- Calculate the distance between every point with the fitting ground h
- If h is larger than a setting threshold, then marke it as an obstacle
- Transfer the Point Cloud data of the obstacle to the robot coordinate space
- Combine the current robot moving stats with the obstacle to decide how to avoid the obstacles

In the traditional way of doing obstacle avoidance task, one should use the camera to detect the point flow to get the information about the environment. For example, the position of the floor and obstacles. Then we can use some fixed way to plan and executes.

- Use Depth camera to get Point Cloud
- Use the Point Cloud to fitting a function about the ground
- Calculate the distance between every point with the fitting ground h
- If h is larger than a setting threshold, then marke it as an obstacle
- Transfer the Point Cloud data of the obstacle to the robot coordinate space
- Combine the current robot moving stats with the obstacle to decide how to avoid the obstacles

Related Work: Use CNN and transfer learning

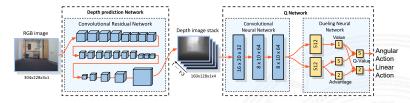


Use transfer learning (Fine tune) is also a good way to improve the training process.

²http://cs231n.stanford.edu/slides/2016/winter1516_lecture11.pdf

Related Work: Using CNN and reinforcement learning

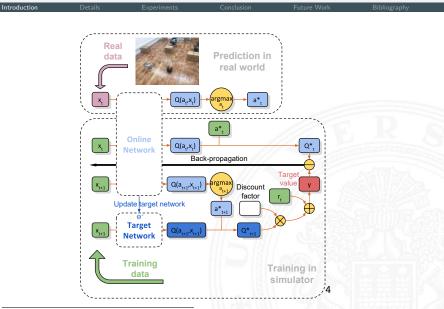
[Xie et al., 2017] Towards Monocular Vision based Obstacle Avoidance through Deep Reinforcement Learning. $^{\rm 3}$



This paper uses CNN to get a "Depth Image stack", then uses D3QN to get action.

³https://arxiv.org/abs/1706.09829

Related Work: Using CNN and reinforcement learning



⁴https://arxiv.org/abs/1706.09829



1. Introduction

2. Details

Try to use CNN Use self supervised learning Use Reinforcement Learning

- 3. Experiments
- 4. Conclusion
- 5. Future Work
- 6. Bibliography





| In | troduction | Details | Experiments | Conclusion | Future Work | Bibliography |
|----|------------|---------|-------------|------------|-------------|--------------|
| | | | | | | |



Pipeline:

- Collect data and labeling
- Build CNN to train
- Get the result in test set



| Introduction | Details | Experiments | Conclusion | Future Work | Bibliography |
|--------------|---------|-------------|------------|-------------|--------------|
| | | | | | |
| | | | | | |



Pipeline:

- Collect data and labeling
- Build CNN to train
- Get the result in test set

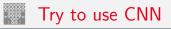


| Introduction | Details | Experiments | Conclusion | Future Work | Bibliography |
|--------------|---------|-------------|------------|-------------|--------------|
| | | | | | |
| | | | | | |
| | | | | | |



Pipeline:

- Collect data and labeling
- Build CNN to train
- Get the result in test set



| | Details | | |
|--|---------|--|--|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

mondi



- Collect data and labeling
- Build CNN to train
- Get the result in test set

| 202 | J | | | J | | |
|-----|--------------|---------------------------|-------------|------------|------------------------------------|--------------|
| | Introduction | Details | Experiments | Conclusion | Future Work | Bibliography |
| | | fc_3→-> in fc_2_→-> in | | | of the CNN lutional Netw | orks |
| | | fc_1fat_ | | | nv 1: filter 16, e 5, strides 1 | kernel |
| | | pool3_ | | | nv 2: filter 32, e 5, strides 1 | kernel |
| | | pool2_ | | | nv 3: filter 54, e 5, strides 1 | kernel |
| | | pool1_ | | ► Fully c | onnected Ne | tworks |
| | | convolution1_ | | | | |
| | | input_ | | | | |

| Introduction | Details | Experiments | Conclusion | Future Work | Bibliography |
|--------------|---|-------------|------------|-------------------------------------|--------------|
| | fc_3_ fc_2_ ↑ → → → → → → → → → → → → → → → → → → → | | | of the CNN | vorks |
| | fc_1 | | | onv 1: filter 16, e 5, strides 1 | , kernel |
| | pool3_ | | | nv 2: filter 32, e 5, strides 1 | kernel |
| | pool2_ | | | nv 3: filter 54, e 5, strides 1 | kernel |
| | pool1_ | | ► Fully | connected Ne | tworks |
| | convolution1> int | | | | |
| | inpu | t_ | | | |

| 22 | J | | | . | | |
|----|--------------|-----------------------------|-------------|------------|------------------------------------|--------------|
| | Introduction | Details | Experiments | Conclusion | Future Work | Bibliography |
| | | fc_3_ fc_2_ ↑+>>> int | | | of the CNN lutional Netw | vorks |
| | | fc_1inst | | | nv 1: filter 16, e 5, strides 1 | kernel |
| | | pool3_ convolution3> ext | | | nv 2: filter 32, e 5, strides 1 | kernel |
| | | pool2_ | | | nv 3: filter 54, e 5, strides 1 | kernel |
| | | pool1_ | | ► Fully o | connected Ne | tworks |
| | | convolution1_ +> == == | | | | |

| <u>88 : </u> | Introduction | Details | Experiments | Conclusion | Future Work | Bibliography |
|--------------|--------------|-----------------------------|-------------|------------|-------------------------------------|--------------|
| | | fc_3++ int fc_2++ int | | | of the CNN | vorks |
| | | fc_1fat_ | | | onv 1: filter 16, e 5, strides 1 | kernel |
| | | pool3_ convolution3>> == | | | onv 2: filter 32, e 5, strides 1 | kernel |
| | | pool2_ | | | onv 3: filter 54, e 5, strides 1 | kernel |
| | | | | ► Fully o | connected Ne | tworks |
| | | convolution1_ >> int | 2 | | | |

| 202 : | J | | | , | | |
|-------|--------------|-----------------------------|-------------|------------|------------------------------------|--------------|
| | Introduction | Details | Experiments | Conclusion | Future Work | Bibliography |
| | | fc_3 int fc_2 int | | | of the CNN lutional Netw | orks |
| | | fc_1fat_ | | | nv 1: filter 16, e 5, strides 1 | kernel |
| | | pool3_ convolution3> int | | | nv 2: filter 32, e 5, strides 1 | kernel |
| | | pool2_ | | | nv 3: filter 54, e 5, strides 1 | kernel |
| | | pool1_ | | ► Fully c | connected Ne | tworks |
| | | convolution1_ | | 📂 Fla | t 1: 100 nodes | |
| | | | * | ► Fla | t 2: 50 nodes | |
| | | input_ | _ | Fla | t 3: 2 nodes | |

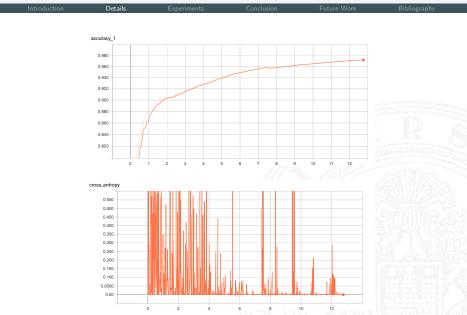
| | | | J | | |
|--------------|------------------------|-------------|------------|-----------------------------------|--------------|
| Introduction | Details | Experiments | Conclusion | Future Work | Bibliography |
| | fc_3>∞ an fc_2>∞ an | | | of the CNN Iutional Netv | vorks |
| | fc_1_ +> int | | | nv 1: filter 16 e 5, strides 1 | , kernel |
| | pool3_ | | | nv 2: filter 32 e 5, strides 1 | , kernel |
| | pool2_ | | | nv 3: filter 54 e 5, strides 1 | , kernel |
| | pool1_ | | ► Fully o | connected Ne | etworks |
| | convolution1_ | | ► Fla | at 1: 100 node | S |
| | | | ► Fla | at 2: 50 nodes | |
| | input | t | Fla | at 3: 2 nodes | |

| Introduction | Details | Experiments | Conclusion | Future Work | Bibliography |
|--------------|---|-------------|------------|------------------------------------|--------------|
| | fc_3_ ↑ → → → int fc_2_ → → → int | | | of the CNN Iutional Netw | vorks |
| | fc_1fat | | | nv 1: filter 16, e 5, strides 1 | kernel |
| | pool3_ | | | nv 2: filter 32, e 5, strides 1 | kernel |
| | pool2_ | | | nv 3: filter 54, e 5, strides 1 | kernel |
| | pool1_ | | ► Fully o | connected Ne | tworks |
| | convolution1_ | | ► Fla | nt 1: 100 node | 5 |
| | | | ► Fla | at 2: 50 nodes | |
| | input | 6 | Fla | t 3: 2 nodes | |

Try to use CNN: Structure of my CNN

| 001 | J | | | J | | | |
|-----|--------------|----------------------------------|-------------|------------|------------------------------------|--------------|--|
| | Introduction | Details | Experiments | Conclusion | Future Work | Bibliography | |
| | | fc_3→+>⊂⇒ int fc_2→+>⊂> int | | | of the CNN Intional Netv | vorks | |
| | | fc_1>> int | | | onv 1: filter 16 e 5, strides 1 | , kernel | |
| | | pool3_ convolution3> == == == | | | onv 2: filter 32 e 5, strides 1 | , kernel | |
| | | pool2_ | | | onv 3: filter 54 e 5, strides 1 | , kernel | |
| | | pool1_ | | ► Fully | connected Ne | etworks | |
| | | convolution1 | | 🕞 Fla | at 1: 100 node | S | |
| | | | | ► Fla | at 2: 50 nodes | | |
| | | inpu | it_ | ► Fla | at 3: 2 nodes | | |





Use self supervised learning



Details

The basic idea of this is to let robot get labels by the additional sensing such as range finder or collision detection switch. But this way is highly relying on the labeling rules.



Use self supervised learning



Details

The basic idea of this is to let robot get labels by the additional sensing such as range finder or collision detection switch. But this way is highly relying on the labeling rules.

Using the bumper in front of the turtlebot



Use self supervised learning



Details

The basic idea of this is to let robot get labels by the additional sensing such as range finder or collision detection switch. But this way is highly relying on the labeling rules.

Using depth sensor to get "ground truth"

Using the bumper in front of the turtlebot



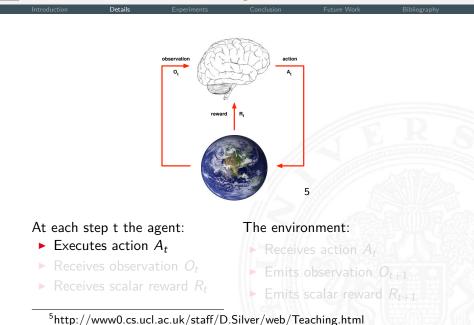




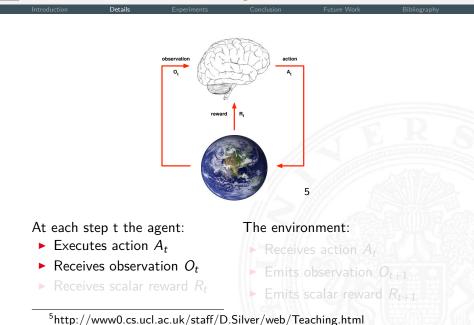
The basic idea of this is to let robot get labels by the additional sensing such as range finder or collision detection switch. But this way is highly relying on the labeling rules.

- Using depth sensor to get "ground truth"
- Using the bumper in front of the turtlebot

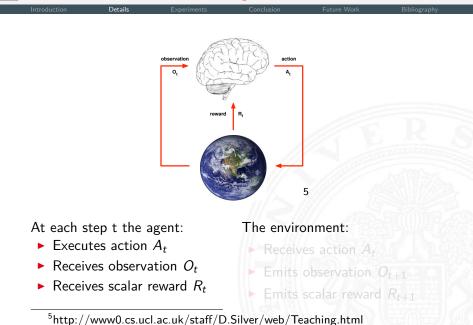




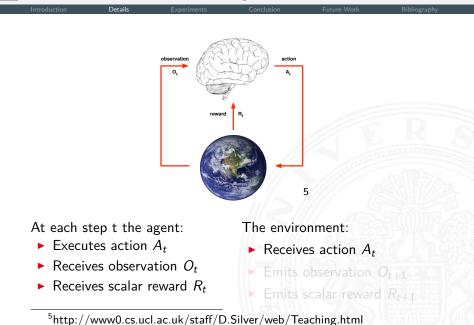
H. Liang – Mobile robot avoid obstascles

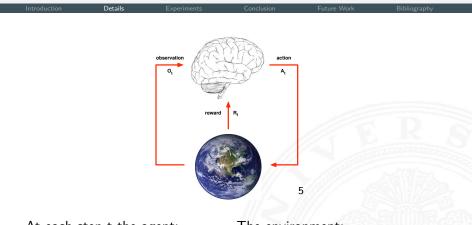


H. Liang - Mobile robot avoid obstascles



H. Liang - Mobile robot avoid obstascles





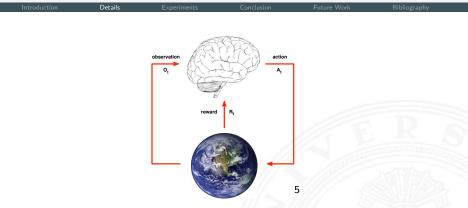
- At each step t the agent:
 - Executes action A_t
 - Receives observation O_t
 - Receives scalar reward R_t

The environment:

- Receives action A_t
- Emits observation O_{t+1}

Emits scalar reward R_{t+1}

⁵http://www0.cs.ucl.ac.uk/staff/D.Silver/web/Teaching.html



At each step t the agent:

- Executes action A_t
- Receives observation O_t
- Receives scalar reward R_t

The environment:

- Receives action A_t
- Emits observation O_{t+1}
- Emits scalar reward R_{t+1}

⁵http://www0.cs.ucl.ac.uk/staff/D.Silver/web/Teaching.html

Use Reinforcement Learning: Q-Learning



The mission of RL is to find a *best policy* in order to make the reward more.

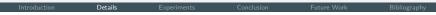
The value function(Bellman Function) describes how to get the value of current state, that is,

$$V(s) = \mathbb{E}[R_{t+1} + \lambda v(S_{t+1})|S_t = s]$$

To update the Q value, that is,

 $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \lambda \max Q(S_{t+1}, a) - Q(S_t, A_t))$

Use Reinforcement Learning: Q-Learning



The mission of RL is to find a *best policy* in order to make the reward more.

The value function(Bellman Function) describes how to get the value of current state, that is,

$$V(s) = \mathbb{E}[R_{t+1} + \lambda v(S_{t+1})|S_t = s]$$

To update the Q value, that is,

 $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \lambda \max Q(S_{t+1}, a) - Q(S_t, A_t))$

Use Reinforcement Learning: Q-Learning



The mission of RL is to find a *best policy* in order to make the reward more.

The value function(Bellman Function) describes how to get the value of current state, that is,

$$V(s) = \mathbb{E}[R_{t+1} + \lambda v(S_{t+1})|S_t = s]$$

To update the Q value, that is,

 $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \lambda \max_{a} Q(S_{t+1}, a) - Q(S_t, A_t))$

Use Reinforcement Learning: DQN

| Introduction | Details | Experiments | Conclusion | Future Work | Bibliography |
|--------------|---------|-------------|------------|-------------|--------------|
| | | | | | |

In the Q-Learning we give a Q table:

| | a ₁ | a ₂ | a ₃ | a ₄ |
|-----------------------|----------------|----------------|----------------|----------------|
| <i>s</i> ₁ | Q(1,1) | Q(1,2) | Q(1,3) | Q(1,4) |
| <i>s</i> ₂ | Q(2,1) | Q(2,2) | Q(2,3) | Q(2,4) |
| <i>s</i> ₃ | Q(3,1) | Q(3,2) | Q(3,3) | Q(3,4) |
| <i>S</i> 4 | Q(4,1) | Q(4,2) | Q(4,3) | Q(4,4) |

Due to store too many states in one Q table is unrealistic in DQN, we get Q table from a network.



Use Reinforcement Learning: DQN

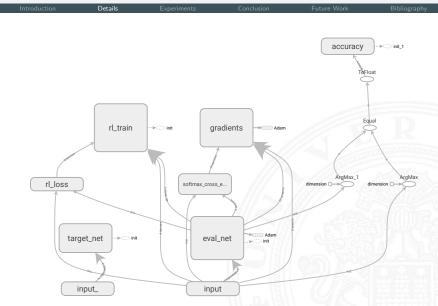
| Introduction | Details | Experiments | Conclusion | Future Work | Bibliography |
|--------------|---------|-------------|------------|-------------|--------------|
| | | | | | |

In the Q-Learning we give a Q table:

| | a ₁ | a ₂ | a ₃ | a ₄ |
|-----------------------|----------------|----------------|----------------|----------------|
| <i>s</i> ₁ | Q(1,1) | Q(1,2) | Q(1,3) | Q(1,4) |
| <i>s</i> ₂ | Q(2,1) | Q(2,2) | Q(2,3) | Q(2,4) |
| <i>s</i> ₃ | Q(3,1) | Q(3,2) | Q(3,3) | Q(3,4) |
| <i>S</i> 4 | Q(4,1) | Q(4,2) | Q(4,3) | Q(4,4) |

Due to store too many states in one Q table is unrealistic in DQN, we get Q table from a network.

Use Reinforcement Learning: DQN Algorithm



Use Reinforcement Learning: Get Reward

```
Details
def get_reward(self, action):
      if self.flag == action:
           if action == 0:
               reward = 1
           else:
               reward = 0
      else:
           reward = -1
      return reward
```

Flag is whether the robot is crashed a bumper. In the future, we need to add more actions.



3

| tro | duction D | etails | Experiments | Conclusion | Future Work | Bibliography |
|-----|----------------------------------|--------|-------------|------------|-------------|--------------|
| | Introductio | n | | | | |
|) | Details | | | | | |
| 3. | Experiment Experim Experim | ient 1 | | | | |
| | Conclusion | | | | | |
| | Future Wo | rk | | | | ananan |
|). | Bibliograph | ıy | | | | all al |
| | | | | | | |



| Introduction | Details | Experiments | Conclusion | Future Work | Bibliography |
|--------------|---------|-------------|------------|-------------|--------------|
| | | | | | |

Experiment using CNN





| | Experiments | | |
|--|-------------|--|--|
| | | | |

Experiment using DQN





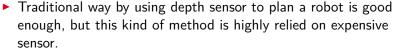
| Intro | duction D | etails | Experiments | Conclusion | Future Work | Bibliography |
|-------|-------------|--------|-------------|------------|-------------|---|
| 1. | Introductio | n | | | | |
| 2. | Details | | | | | |
| 3. | Experimen | | | | | |
| 4. | Conclusion | | | | | |
| 5. | Future Wo | rk | | | | n |
| 6. | Bibliograph | ıy | | | | K S |
| | | | | | | And |
| | | | | | | A 1972a |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | 2. 2. 202 |
| | | | | | | |
| | | | | | | 新田 |



- Traditional way by using depth sensor to plan a robot is good enough, but this kind of method is highly relied on expensive sensor.
- By using CNN the Monocular camera can get very good result in both training data and test data. But it will get bad result in unknown environment.
- Self-supervised learning can reduce the backward of the pure CNN, by self collecting data on the run.
- By using CNN to get a predict depth image then use this as input to train the RL network is very interesting, and it avoid the backward of using bad simulated camera in simulation environment.
- Using DQN to train the network directly is more intelligence, need to do Some furthermore researching on it.

By using CNN the Monocular camera can get very good result in both training data and test data. But it will get bad result in unknown environment.

- Self-supervised learning can reduce the backward of the pure CNN, by self collecting data on the run.
- By using CNN to get a predict depth image then use this as input to train the RL network is very interesting, and it avoid the backward of using bad simulated camera in simulation environment.
- Using DQN to train the network directly is more intelligence, need to do Some furthermore researching on it.



Conclusion

24 / 30



Conclusion

Using DQN to train the network directly is more intelligence.

- By using CNN to get a predict depth image then use this as
- Self-supervised learning can reduce the backward of the pure CNN, by self collecting data on the run.
- By using CNN the Monocular camera can get very good result in both training data and test data. But it will get bad result in unknown environment.
- enough, but this kind of method is highly relied on expensive sensor.
- Conclusion Traditional way by using depth sensor to plan a robot is good

unknown environment.

Self-supervised learning can reduce the backward of the pure CNN, by self collecting data on the run. By using CNN to get a predict depth image then use this as

- input to train the RL network is very interesting, and it avoid the backward of using bad simulated camera in simulation environment.
- Using DQN to train the network directly is more intelligence.

sensor.

By using CNN the Monocular camera can get very good result in both training data and test data. But it will get bad result in

Traditional way by using depth sensor to plan a robot is good enough, but this kind of method is highly relied on expensive



Self-supervised learning can reduce the backward of the pure CNN, by self collecting data on the run. By using CNN to get a predict depth image then use this as input to train the RL network is very interesting, and it avoid

- the backward of using bad simulated camera in simulation environment.
- Using DQN to train the network directly is more intelligence, need to do Some furthermore researching on it.

- enough, but this kind of method is highly relied on expensive sensor.
- Traditional way by using depth sensor to plan a robot is good

By using CNN the Monocular camera can get very good result in both training data and test data. But it will get bad result in

Conclusion

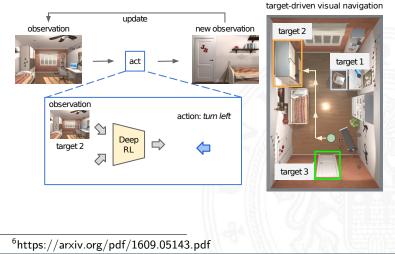
unknown environment.



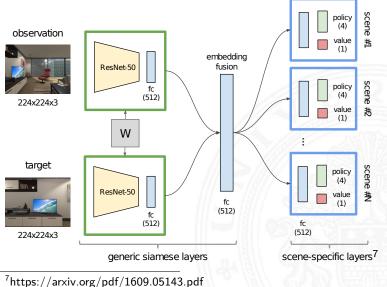
| Intro | duction De | tails | Experiments | Conclusion | Future Work | Bibliography | |
|-------|-------------|-------|-------------|------------|-------------|--------------|-------|
| 1. | Introductio | n | | | | | |
| 2. | Details | | | | | | |
| 3. | Experiment | | | | | | |
| 4. | Conclusion | | | | | | |
| 5. | Future Wo | rk | | | | | |
| 6. | Bibliograph | ıy | | | | | S |
| | | | | | | | 222 |
| | | | | | | | a |
| | | | | | | | E |
| | | | | | | | 23 |
| | | | | | | | 122.9 |
| | | | | | | | 231 |
| | | | | | | | |
| | | | | | | | 68 |



[Zhu et al., 2016] Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning $^{\rm 6}$







Future Work



| Introduction Details | Experiments | Conclusion | Future Work | Bibliography |
|----------------------|-------------|------------|-------------|--------------|
| 1. Introduction | | | | |
| 2. Details | | | | |
| 3. Experiments | | | | |
| 4. Conclusion | | | | |
| 5. Future Work | | | | |
| 6. Bibliography | | | | K S |
| | | | | omannesses |
| | | | | 58 9 7 3 a |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | 日田田と |
| | | | | 四時間 |



[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

[Xie et al., 2017] Xie, L., Wang, S., Markham, A., and Trigoni, N. (2017). Towards Monocular Vision based Obstacle Avoidance through Deep Reinforcement Learning.

[Zhu et al., 2016] Zhu, Y., Mottaghi, R., Kolve, E., Lim, J. J., Gupta, A., Fei-Fei, L., and Farhadi, A. (2016).

Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning.

In Robotics and Automation (ICRA), 2017 IEEE International Conference on, pages 3357–3364. IEEE.



[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

[Xie et al., 2017] Xie, L., Wang, S., Markham, A., and Trigoni, N. (2017). Towards Monocular Vision based Obstacle Avoidance through Deep Reinforcement Learning.

[Zhu et al., 2016] Zhu, Y., Mottaghi, R., Kolve, E., Lim, J. J., Gupta, A. Fei-Fei, L., and Farhadi, A. (2016).

Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning.

In Robotics and Automation (ICRA), 2017 IEEE International Conference on, pages 3357–3364. IEEE.



[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

[Xie et al., 2017] Xie, L., Wang, S., Markham, A., and Trigoni, N. (2017). Towards Monocular Vision based Obstacle Avoidance through Deep Reinforcement Learning.

[Zhu et al., 2016] Zhu, Y., Mottaghi, R., Kolve, E., Lim, J. J., Gupta, A., Fei-Fei, L., and Farhadi, A. (2016).

Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning.

In Robotics and Automation (ICRA), 2017 IEEE International Conference on, pages 3357–3364. IEEE.

Thank you for your attention, Suggestion, Questions and Commands are highly welcome.

TA

MS