**1. Purpose and Scope**

**1.1 Purpose**  
Define end‑to‑end requirements for a direction‑sensitive, obstacle‑optimized spiking neural network (SNN) optical‑flow system with real‑time, embedded deployment.

**1.2 Scope**  
Covers environment setup, data preparation, model training, inference pipeline, post‑processing, deployment optimizations, and verification tests.

**2. Environment & Dependencies**

**2.1 Workspace Structure**

* Base directory: ~/snn\_optical\_flow/
  + data/
  + models/
  + scripts/

**2.2 CUDA & GPU Toolkit**

* CUDA version: **11.3.1**
* Installer: cuda\_11.3.1\_465.19.01\_linux.run

**2.3 Python Environment**

* Create virtualenv named oflow\_env (Python 3.x).
* Activate with source oflow\_env/bin/activate.

**2.4 Core Python Dependencies**

torch==1.12.1+cu113

torchvision==0.13.1+cu113

numpy==1.23.5

pandas==1.5.3

spikingjelly==0.0.0.0.13

opencv-python==4.7.0.72

imageio==2.27.0

numba==0.57.0

h5py==3.8.0

hdf5plugin==4.3.0

tqdm==4.65.0

Install via:

pip3 install torch==1.12.1+cu113 torchvision==0.13.1+cu113 --extra-index-url https://download.pytorch.org/whl/cu113

pip3 install -r requirements.txt

**3. Data Preparation**

**3.1 Source Dataset Structure**

data/

└── dsec/

├── train\_forward/

│ ├── events/left/rectified/

│ └── disparity/

├── val\_forward/

│ ├── events/left/rectified/

│ └── disparity/

└── test\_forward/

├── events/left/rectified/

└── disparity/

**3.2 Conversion to Repository Format**

* Target structure:  
  data/dataset/raw\_files/{train\_forward,val\_forward,test\_forward}
* Copy only events/left/rectified/ into each split folder.
* Example script (to run in project root):
* python3 - <<END
* from pathlib import Path
* import shutil
* dsec\_root = Path('data/dsec')
* repo\_root = Path('data/dataset/raw\_files')
* for split in ['train\_forward','val\_forward','test\_forward']:
* src = dsec\_root/f"{split}/events/left/rectified"
* dst = repo\_root/f"{split}"
* dst.mkdir(parents=True, exist\_ok=True)
* shutil.copytree(src, dst, dirs\_exist\_ok=True)
* END

**4. Model Training**

**4.1 Model Architecture**

* Base class: ThreeDNetCat1Res
* Obstacle‑aware hyperparameters:
  + temporal\_kernel\_size = 7
  + angular\_weight = 0.85
  + floor\_mask = True

**4.2 Training Configuration**

* Optimizer: AdamW (learning rate 2 × 10⁻⁴, weight\_decay 1 × 10⁻⁵)
* Scheduler: CosineAnnealingLR (T\_max = 50)
* Command example:
* python train\_obstacle.py \
* --batch\_size 8 \
* --temporal\_window 21 \
* --max\_epochs 100 \
* --flow\_threshold 0.4 \
* --obstacle\_weight 0.75

**5. Real‑Time Inference Pipeline**

**5.1 Model Loading & Precision**

* Load with TorchScript: torch.jit.load('models/obstacle\_optimized.pt')
* Convert to half‑precision .half() and move to CUDA.

**5.2 Processing Loop**

1. Acquire event batch (~9 ms intervals)
2. Preprocess to tensor → .half() → .cuda()
3. with torch.no\_grad(): flow = model(tensor)
4. Reset network state: spikingjelly.activation\_based.functional.reset\_net(model)
5. Convert flow to obstacle map (see Section 6)
6. Display or forward the map

**6. Obstacle Detection Post‑Processing**

**6.1 Flow ⇒ Obstacle Map**

* Input: flow\_tensor shape [B, 2, H, W]
* Compute:
  + magnitude = torch.norm(flow\_tensor, dim=1)
  + direction = torch.atan2(flow\_tensor[:,1], flow\_tensor[:,0])

**6.2 Dynamic Thresholding**

threshold = 0.25 \* magnitude.mean() + 0.15

obstacle\_mask = (magnitude > threshold).float()

**6.3 Directional Filtering**

approach\_mask = (direction.abs() < np.pi/4).float()

obstacle\_map = obstacle\_mask \* approach\_mask

**7. Deployment Optimizations**

**7.1 TensorRT Conversion**

python3 -m torch2trt models/obstacle\_optimized.pt \

--fp16 \

--input-size 1,11,260,346 \

--batch-size 1 \

-o models/engine.trt

**7.2 Performance Verification**

trtexec --loadEngine=models/engine.trt \

--shapes=1x11x260x346 \

--fp16 \

--verbose

**8. Verification & Testing**

**8.1 Unit Tests** (test\_obstacle\_avoidance.py)

* **Emergency Stop Test**
  + Generate synthetic approaching flow (e.g. velocity=5.0)
  + Assert response['brake'] > 0.8
* **Lateral Avoidance Test**
  + Generate lateral flow (angle = π/3)
  + Assert response['steer'] != 0

**9. Key Configuration Parameters**

| **Parameter** | **Obstacle‑Tuned** | **Standard** | **Description** |
| --- | --- | --- | --- |
| Temporal Window | 21 frames | 11 frames | Motion history depth |
| Angular Weight | 0.85 | 0.50 | Directional emphasis |
| Flow Threshold | 0.4 px/frame | 0.2 px/frame | Obstacle sensitivity |
| Floor Masking | Enabled | Disabled | Ground‑plane motion removal |
| Quantization | FP16 | FP32 | Inference speed & size trade‑off |

**10. Non‑Functional Requirements**

* **Latency:** ≤ 10 ms per batch
* **Throughput:** ≥ 100 Hz event processing
* **GPU Memory:** ≤ 4 GB
* **CPU Load:** ≤ 10% during inference

**11. Next Steps**

1. Integrate into Cursor AI as a detailed prompt.
2. Automate CI/CD pipelines for setup, training, and testing.
3. Expand README with usage examples and benchmarks.

*End of Document*