optimizationWithImage_working_moving_point

June 26, 2023

[ ]:
```
import numpy as np
import os
import pathlib
import random
import torch
import torchgeometry
import cv2
from dlclive import DLCLive, Processor

import matplotlib.pyplot as plt
from random import randrange
import random
from IPython.display import clear_output

from torch.utils.data import DataLoader

# potential mass parametrizations
from differentiable_robot_model.rigid_body_params import (
    UnconstrainedScalar,
    PositiveScalar,
    UnconstrainedTensor,
)

# potential inertia matrix parametrizations
from differentiable_robot_model.rigid_body_params import (
    CovParameterized3DInertiaMatrixNet,
    Symm3DInertiaMatrixNet,
    SymmPosDef3DInertiaMatrixNet,
    TriangParam3DInertiaMatrixNet,
)

from differentiable_robot_model.robot_model import (
    DifferentiableRobotModel,
    DifferentiableKUKAiiwa,
)
from differentiable_robot_model.data_utils import (
    generate_sine_motion_forward_dynamics_data,
```

```
)
import diff_robot_data

torch.set_printoptions(precision=3, sci_mode=False)

random.seed(0)
np.random.seed(1)
torch.manual_seed(0)
```

[ ]:
```
<torch._C.Generator at 0x7fe1287d89b0>
```

[ ]:
```
class NMSELoss(torch.nn.Module):
    def __init__(self, var):
        super(NMSELoss, self).__init__()
        self.var = var

    def forward(self, yp, yt):
        err = (yp - yt) ** 2
        werr = err / self.var
        return werr.mean()

class ConstrainedTensor(torch.nn.Module):
    def __init__(self, dim1, dim2, init_tensor=None, init_std=0.1, min_val=0.0,␣
 ↪max_val=1.0):
        super().__init__()
        self._dim1 = dim1
        self._dim2 = dim2
        if init_tensor is None:
            init_tensor = torch.empty(dim1, dim2).normal_(mean=0.0,␣
 ↪std=init_std)
        self.param = torch.nn.Parameter(init_tensor)
        self.min_val = min_val
        self.max_val = max_val

    def forward(self):
        param = self.param
        param = torch.clamp(param, min=self.min_val, max=self.max_val)
        return param

urdf_path = os.path.join("a1.urdf")
device = "cpu"
learnable_robot_model = DifferentiableRobotModel(
    urdf_path, "A1", device=device
)

learnable_robot_model.make_link_param_learnable(
```

```
    "FR_hip", "trans", ConstrainedTensor(dim1 = 1, dim2 = 3, min_val = 0,␣
 ↪max_val = 10)
)
learnable_robot_model.make_link_param_learnable(
        "FR_thigh_shoulder", "trans", ConstrainedTensor(dim1 = 1, dim2 = 3,␣
 ↪min_val = 0, max_val = 10)
    )
learnable_robot_model.make_link_param_learnable(
        "FR_thigh", "trans", ConstrainedTensor(dim1 = 1, dim2 = 3, min_val = 0,␣
 ↪max_val = 10)
    )
learnable_robot_model.make_link_param_learnable(
        "FR_calf", "trans", ConstrainedTensor(dim1 = 1, dim2 = 3, min_val = 0,␣
 ↪max_val = 10)
    )
learnable_robot_model.make_link_param_learnable(
        "FR_foot", "trans", ConstrainedTensor(dim1 = 1, dim2 = 3, min_val = 0,␣
 ↪max_val = 10)
    )
learnable_robot_model.make_link_param_learnable(
        "FL_hip", "trans", ConstrainedTensor(dim1 = 1, dim2 = 3, min_val = 0,␣
 ↪max_val = 10)
    )
learnable_robot_model.make_link_param_learnable(
        "FL_thigh_shoulder", "trans", ConstrainedTensor(dim1 = 1, dim2 = 3,␣
 ↪min_val = 0, max_val = 10)
    )
learnable_robot_model.make_link_param_learnable(
        "FL_thigh", "trans", ConstrainedTensor(dim1 = 1, dim2 = 3, min_val = 0,␣
 ↪max_val = 10)
    )
learnable_robot_model.make_link_param_learnable(
        "FL_calf", "trans", ConstrainedTensor(dim1 = 1, dim2 = 3, min_val = 0,␣
 ↪max_val = 10)
    )
learnable_robot_model.make_link_param_learnable(
        "FL_foot", "trans", ConstrainedTensor(dim1 = 1, dim2 = 3, min_val = 0,␣
 ↪max_val = 10)
    )
learnable_robot_model.make_link_param_learnable(
        "RR_hip", "trans", ConstrainedTensor(dim1 = 1, dim2 = 3, min_val = 0,␣
 ↪max_val = 10)
    )
learnable_robot_model.make_link_param_learnable(
        "RR_thigh_shoulder", "trans", ConstrainedTensor(dim1 = 1, dim2 = 3,␣
 ↪min_val = 0, max_val = 10)
```

```
    )
learnable_robot_model.make_link_param_learnable(
        "RR_thigh", "trans", ConstrainedTensor(dim1 = 1, dim2 = 3, min_val = 0,␣
 ↪max_val = 10)
    )
learnable_robot_model.make_link_param_learnable(
        "RR_calf", "trans", ConstrainedTensor(dim1 = 1, dim2 = 3, min_val = 0,␣
 ↪max_val = 10)
    )
learnable_robot_model.make_link_param_learnable(
        "RR_foot", "trans", ConstrainedTensor(dim1 = 1, dim2 = 3, min_val = 0,␣
 ↪max_val = 10)
    )
learnable_robot_model.make_link_param_learnable(
        "RL_hip", "trans", ConstrainedTensor(dim1 = 1, dim2 = 3, min_val = 0,␣
 ↪max_val = 10)
    )
learnable_robot_model.make_link_param_learnable(
        "RL_thigh_shoulder", "trans", ConstrainedTensor(dim1 = 1, dim2 = 3,␣
 ↪min_val = 0, max_val = 10)
    )
learnable_robot_model.make_link_param_learnable(
        "RL_thigh", "trans", ConstrainedTensor(dim1 = 1, dim2 = 3, min_val = 0,␣
 ↪max_val = 10)
    )
learnable_robot_model.make_link_param_learnable(
        "RL_calf", "trans", ConstrainedTensor(dim1 = 1, dim2 = 3, min_val = 0,␣
 ↪max_val = 10)
    )
learnable_robot_model.make_link_param_learnable(
        "RL_foot", "trans", ConstrainedTensor(dim1 = 1, dim2 = 3, min_val = 0,␣
 ↪max_val = 10)
    )

# learnable_robot_model.print_learnable_params()
joint_angles = torch.rand((1, 12), requires_grad=True)
learnable_robot_model.compute_forward_kinematics(joint_angles,"FR_foot")
```

```
Unknown tag "hardwareInterface" in /robot[@name='a1']/transmission[@name='FR_hip
_tran']/actuator[@name='FR_hip_motor']
Unknown tag "hardwareInterface" in /robot[@name='a1']/transmission[@name='FR_thi
gh_tran']/actuator[@name='FR_thigh_motor']
Unknown tag "hardwareInterface" in /robot[@name='a1']/transmission[@name='FR_cal
f_tran']/actuator[@name='FR_calf_motor']
Unknown tag "hardwareInterface" in /robot[@name='a1']/transmission[@name='FL_hip
_tran']/actuator[@name='FL_hip_motor']
Unknown tag "hardwareInterface" in /robot[@name='a1']/transmission[@name='FL_thi
```

```
gh_tran']/actuator[@name='FL_thigh_motor']
Unknown tag "hardwareInterface" in /robot[@name='a1']/transmission[@name='FL_cal
f_tran']/actuator[@name='FL_calf_motor']
Unknown tag "hardwareInterface" in /robot[@name='a1']/transmission[@name='RR_hip
_tran']/actuator[@name='RR_hip_motor']
Unknown tag "hardwareInterface" in /robot[@name='a1']/transmission[@name='RR_thi
gh_tran']/actuator[@name='RR_thigh_motor']
Unknown tag "hardwareInterface" in /robot[@name='a1']/transmission[@name='RR_cal
f_tran']/actuator[@name='RR_calf_motor']
Unknown tag "hardwareInterface" in /robot[@name='a1']/transmission[@name='RL_hip
_tran']/actuator[@name='RL_hip_motor']
Unknown tag "hardwareInterface" in /robot[@name='a1']/transmission[@name='RL_thi
gh_tran']/actuator[@name='RL_thigh_motor']
Unknown tag "hardwareInterface" in /robot[@name='a1']/transmission[@name='RL_cal
f_tran']/actuator[@name='RL_calf_motor']

Warning: No dynamics information for link: base, setting all inertial properties
to 1.
Warning: No dynamics information for link: trunk, setting all inertial
properties to 1.
Warning: No dynamics information for link: imu_link, setting all inertial
properties to 1.
Warning: No dynamics information for link: FR_hip, setting all inertial
properties to 1.
Warning: No dynamics information for link: FR_thigh_shoulder, setting all
inertial properties to 1.
Warning: No dynamics information for link: FR_thigh, setting all inertial
properties to 1.
Warning: No dynamics information for link: FR_calf, setting all inertial
properties to 1.
Warning: No dynamics information for link: FR_foot, setting all inertial
properties to 1.
Warning: No dynamics information for link: FL_hip, setting all inertial
properties to 1.
Warning: No dynamics information for link: FL_thigh_shoulder, setting all
inertial properties to 1.
Warning: No dynamics information for link: FL_thigh, setting all inertial
properties to 1.
Warning: No dynamics information for link: FL_calf, setting all inertial
properties to 1.
Warning: No dynamics information for link: FL_foot, setting all inertial
properties to 1.
Warning: No dynamics information for link: RR_hip, setting all inertial
properties to 1.
Warning: No dynamics information for link: RR_thigh_shoulder, setting all
inertial properties to 1.
Warning: No dynamics information for link: RR_thigh, setting all inertial
properties to 1.
```

```
Warning: No dynamics information for link: RR_calf, setting all inertial
properties to 1.
Warning: No dynamics information for link: RR_foot, setting all inertial
properties to 1.
Warning: No dynamics information for link: RL_hip, setting all inertial
properties to 1.
Warning: No dynamics information for link: RL_thigh_shoulder, setting all
inertial properties to 1.
Warning: No dynamics information for link: RL_thigh, setting all inertial
properties to 1.
Warning: No dynamics information for link: RL_calf, setting all inertial
properties to 1.
Warning: No dynamics information for link: RL_foot, setting all inertial
properties to 1.
```

[ ]:
```
(tensor([[ 0.062,  0.145, -0.043]], grad_fn=<AddBackward0>),
 tensor([[0.315, 0.360, 0.131, 0.868]], grad_fn=<CopySlices>))
```

[ ]:
```
keypoints = """End of Neck
Shoulder
FL_Knee
FL_Ankle
FL_White_TapeTop
FL_White_TapeBot
FR_Knee
FR_Ankle
FL_Red_TapeTop
FL_Red_TapeBot
End of Tail
Hip
BL_Knee
BL_Ankle
BL_Red_TapeTop
BL_Red_TapeBot
BR_Knee
BR_Ankle
BR_Red_TapeTop
BR_Red_TapeBot""".split("\n")
training_keypoints = ['FL_Ankle','FL_Knee','BL_Ankle','BL_Knee']
indices_keypoints_training = [keypoints.index(val) for val in␣
 ↪training_keypoints]

base_dir = pathlib.Path.cwd()
img_path = base_dir / "HorseInferenceFiles/img0088.png"
dlc_model_path = base_dir/"HorseInferenceFiles/
 ↪DLC_HorseProject1_efficientnet-b0_iteration-0_shuffle-1"
image = cv2.imread(str(img_path))
```

```
dlc_proc = Processor()
dlc_live = DLCLive(model_path=str(dlc_model_path), processor=dlc_proc)

out = list()
image_paths = ["img0088.png","img0228.png","img0288.png","img0484.png"]
images = [cv2.imread(str(base_dir / ("HorseInferenceFiles/"+filePath))) for␣
 ↪filePath in image_paths ]
batch_size = len(image_paths)
for ind, image in enumerate(images):
    pose = None
    if ind == 0:
        pose = (dlc_live.init_inference(image)[indices_keypoints_training][:,0:
 ↪2])
    else:
        pose = (dlc_live.get_pose(image)[indices_keypoints_training][:,0:2])
    pose = torch.from_numpy(pose)
    out.append(pose)

# dlc_live.init_inference(image)
# img_keypoints = dlc_live.get_pose(image)

# training_data = img_keypoints[indices_keypoints_training]
# training_data = training_data[:,0:2]
# training_data

training_data = out = torch.stack(out)
training_data
```

[ ]:
```
tensor([[[275.799, 244.884],
         [249.626, 216.098],
         [340.429, 246.946],
         [349.002, 210.681]],

        [[206.467, 256.802],
         [215.914, 218.328],
         [366.530, 252.797],
         [373.324, 215.444]],

        [[249.048, 258.187],
         [243.911, 220.433],
         [410.155, 250.348],
         [389.190, 213.207]],

        [[252.591, 246.068],
         [225.266, 209.619],
         [339.606, 259.536],
         [349.615, 220.834]]])
```

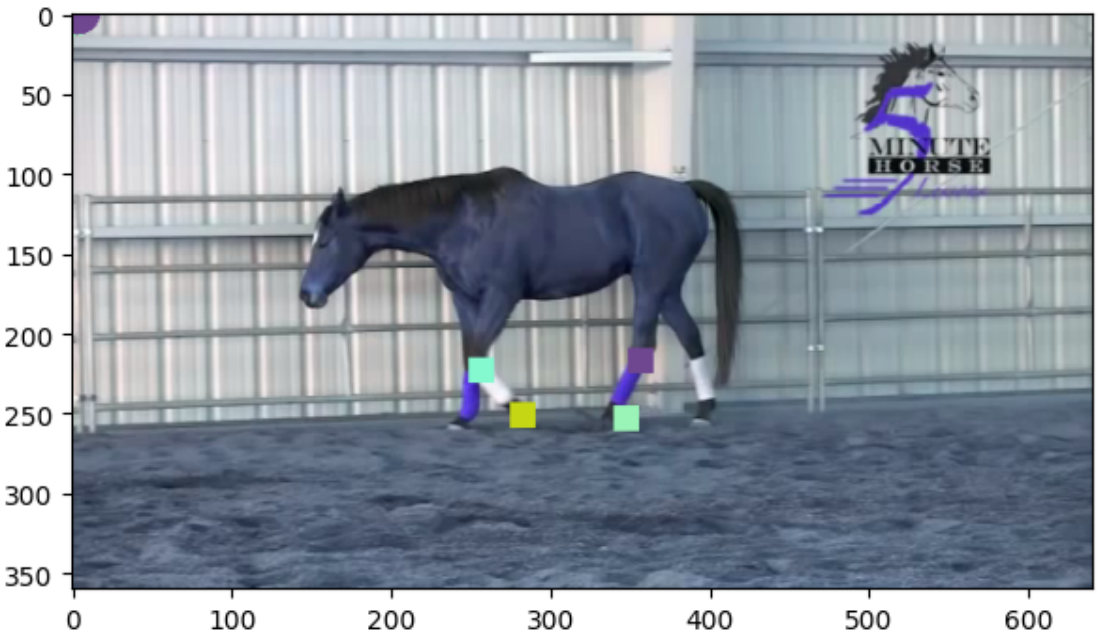
[ ]:
```
def getRobotPositionsFromJointAngles(joint_angles_array, learnable_robot_model):
    """
        Takes in joint angles array and the robot model. Outputs the 3d␣
 ↪position of selected joints of the robot given the joint angles.
        Supports batches of joint angles.
    """
    projections = []
    #Where the Robot Thinks It Is
    for joint_angles in joint_angles_array:
        projection = torch.cat((learnable_robot_model.
 ↪compute_forward_kinematics(joint_angles, "FL_foot")[0],\
        learnable_robot_model.compute_forward_kinematics(joint_angles,␣
 ↪"FL_calf")[0],\
        learnable_robot_model.compute_forward_kinematics(joint_angles,␣
 ↪"RR_foot")[0],\
        learnable_robot_model.compute_forward_kinematics(joint_angles,␣
 ↪"RR_calf")[0]))
        projection = projection.unsqueeze(0).unsqueeze(0)
        projections.append(projection)
    return torch.cat(projections)

def getRobotPositionsInPixelSpace(conversionMatrix, robotPositions):
    """
        Takes in robot positions in 3D space relative to the robot. Then uses␣
 ↪the conversion Matrix to figure out where the robot is in the pixel space of␣
 ↪the camera. Supports batches of robotPositions.
    """
    projectedConversionMatrix = torch.cat([conversionMatrix.unsqueeze(0) for _␣
 ↪in range(batch_size)])
    return torchgeometry.cam2pixel(robotPositions, projectedConversionMatrix)

def getRobotPositionInCamera(joint_angles_array, conversionMatrix,␣
 ↪learnable_robot_model):
    return getRobotPositionsInPixelSpace(conversionMatrix,␣
 ↪getRobotPositionsFromJointAngles(joint_angles_array, learnable_robot_model))
```

[ ]:
```
joint_angles_array = (4*torch.rand((batch_size, 1, 12)))-2
joint_angles_array = joint_angles_array.requires_grad_(True)
conversionMatrix  = (8*torch.randn((4,4)))-4
conversionMatrix = conversionMatrix.requires_grad_(True)
%matplotlib inline
VERBOSE = False

optimizer = torch.optim.Adam([joint_angles_array,conversionMatrix] +␣
 ↪list(learnable_robot_model.parameters()), lr=1e-3)
def debugPrint(*args, **kwargs):
```

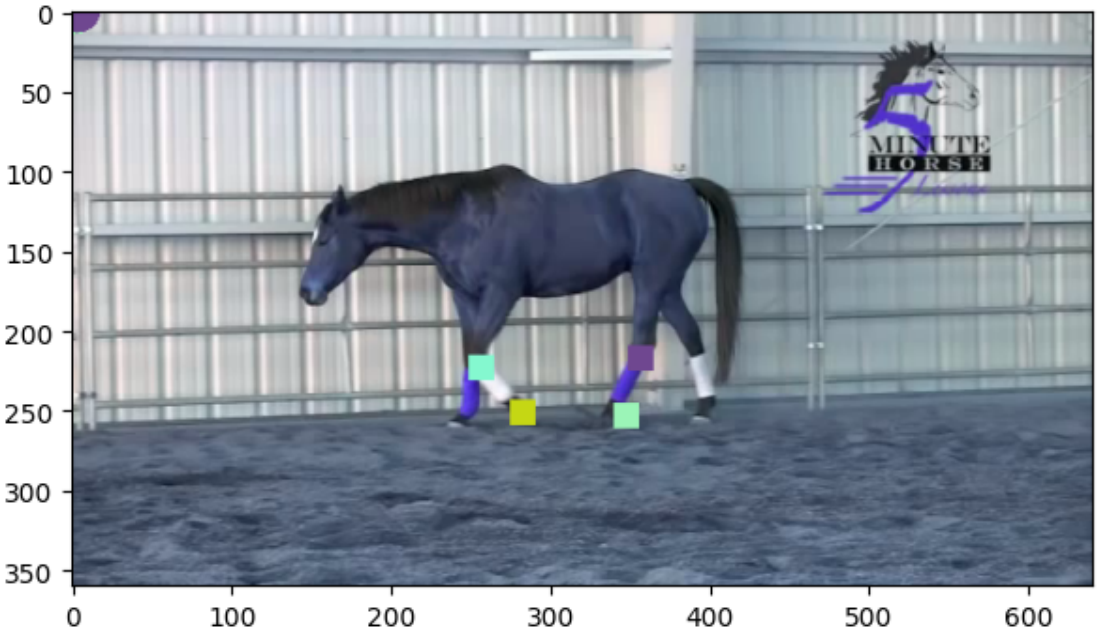
```
    if VERBOSE:
        print(*args, **kwargs)
for epoch in range(2000):
    optimizer.zero_grad()
    cameraEstimate = getRobotPositionInCamera(joint_angles_array,␣
 ↪conversionMatrix, learnable_robot_model)
    debugPrint(f"Camera Estimate Shape: {cameraEstimate.shape}")

    #Loss Between Where Robot/Horse Should Be and Where It Is
    loss = torch.sum(torch.sqrt(torch.sum(torch.
 ↪pow((training_data)-cameraEstimate, 2))))









    if VERBOSE:
        print(f"Epoch {epoch}: Loss ({loss})")
    if epoch % 10 == 0:
        print(f"Epoch {epoch}: Loss ({loss})")

        imgIndex = 0
        img = images[imgIndex].copy()
        random.seed(0)
        for x,y in training_data[imgIndex]:
            color = (randrange(0,256),randrange(0,256),randrange(0,256))
            img = cv2.rectangle(img, (int(x), int(y)),(int(x)+15, int(y)+15),␣
 ↪color, -1)
        random.seed(0)
        for x,y in cameraEstimate[imgIndex][0]:
            color = (randrange(0,256),randrange(0,256),randrange(0,256))
            img = cv2.circle(img, (int(x), int(y)), 15, color, -1)

        plt.clf()
        plt.figure(figsize=(7,5))
        plt.imshow(img)
        plt.show()
    loss.backward()
    optimizer.step()
```

```
Epoch 0: Loss (3085.930419921875)
```

```
<Figure size 640x480 with 0 Axes>
```

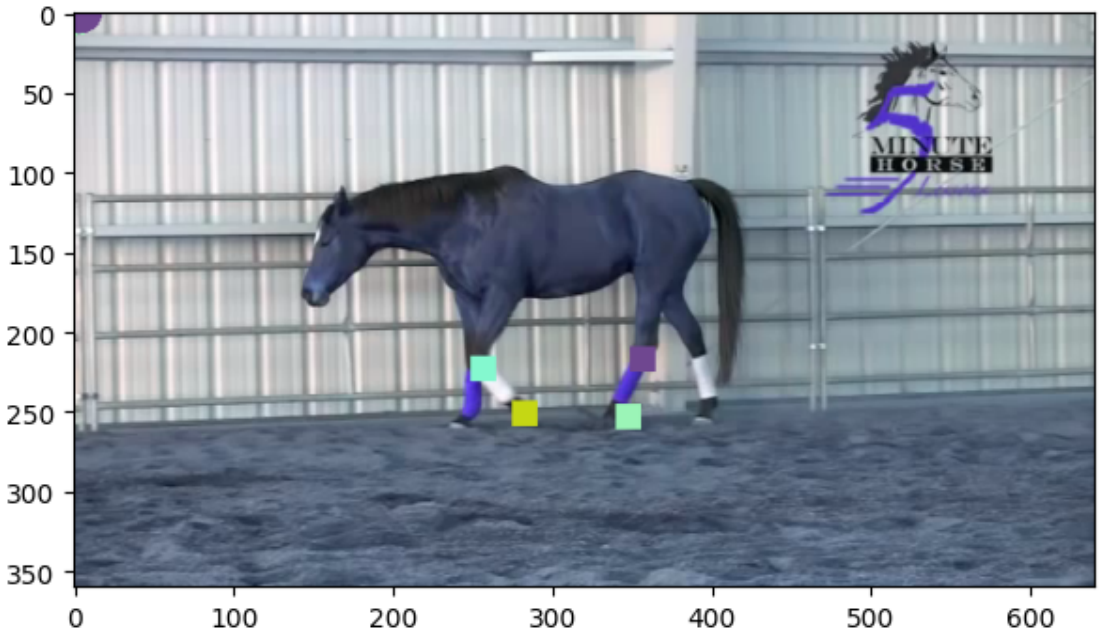



```
Epoch 10: Loss (3084.17822265625)
<Figure size 640x480 with 0 Axes>
```



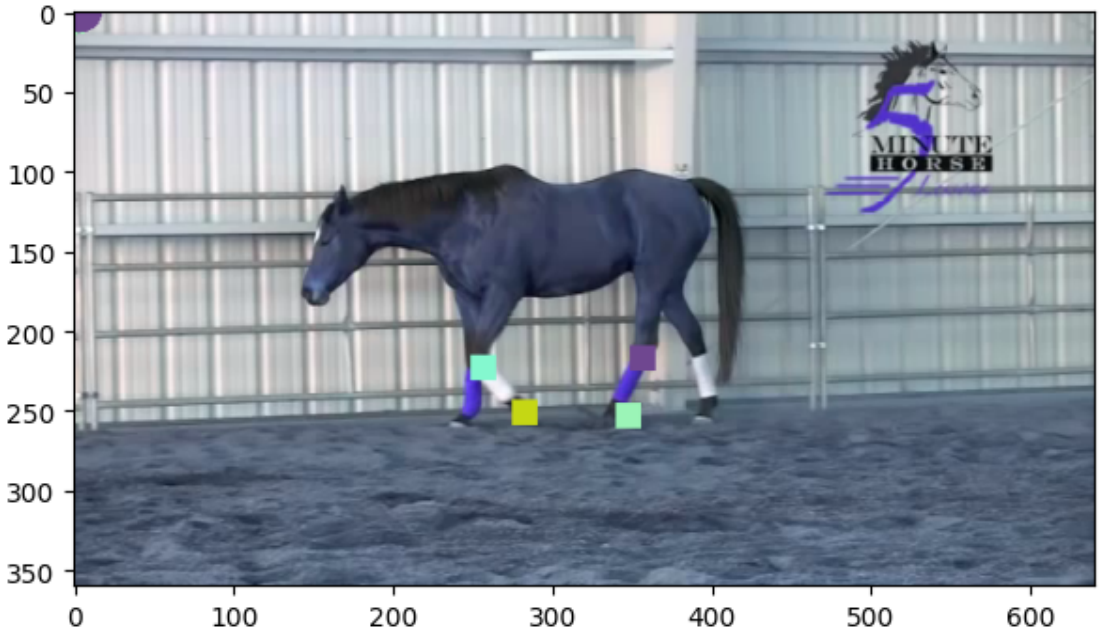

```
Epoch 20: Loss (3081.4423828125)
```

<Figure size 640x480 with 0 Axes>



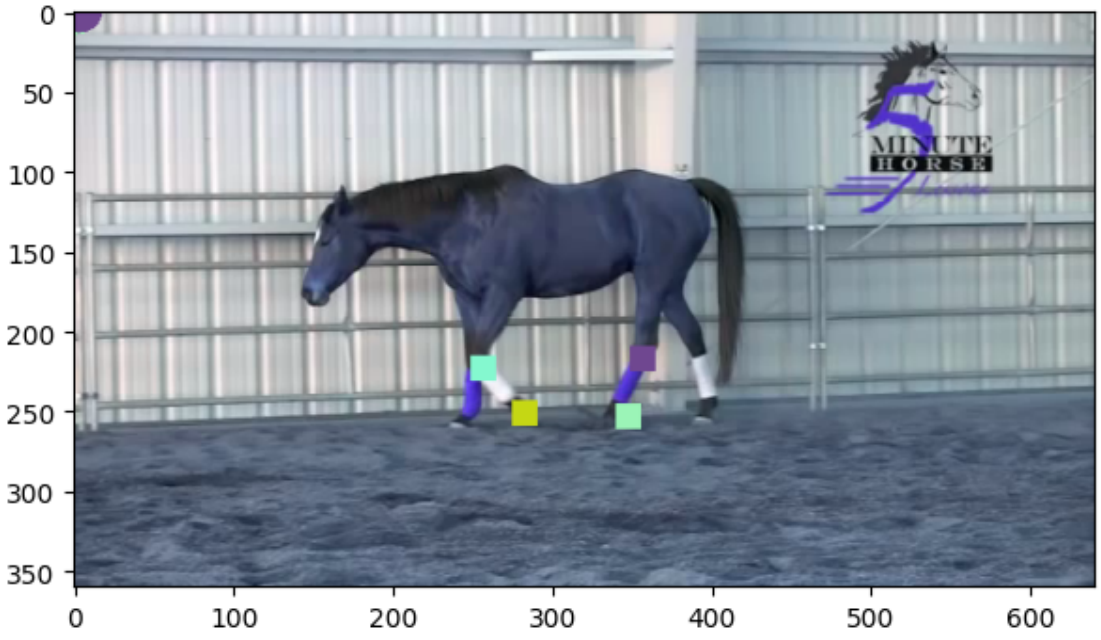
Epoch 30: Loss (3077.74169921875)

<Figure size 640x480 with 0 Axes>



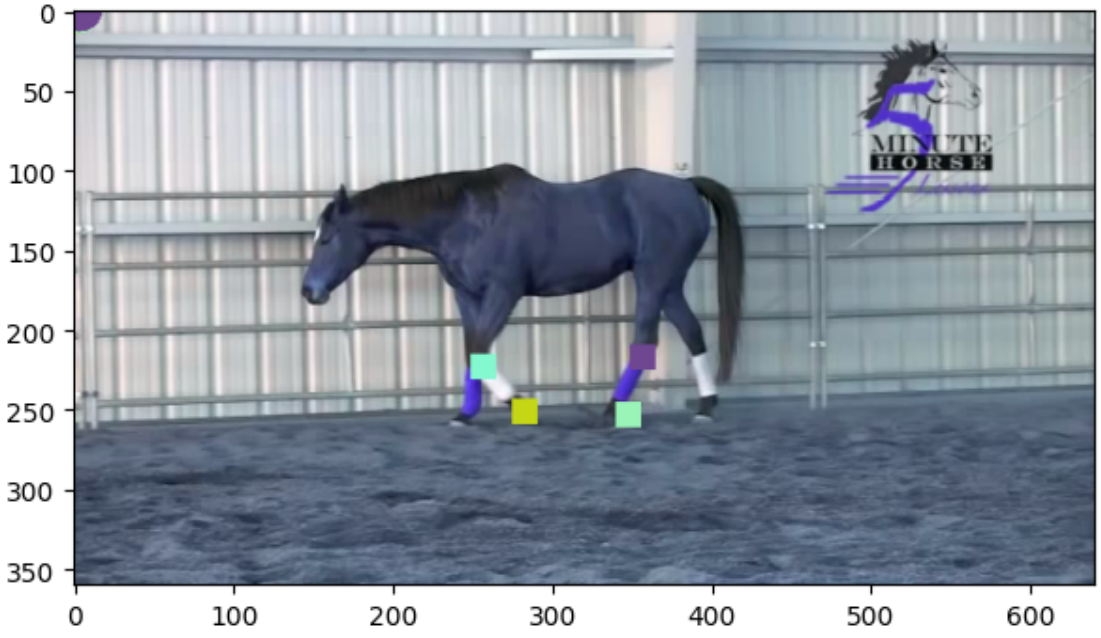
Epoch 40: Loss (3074.660888671875)

<Figure size 640x480 with 0 Axes>

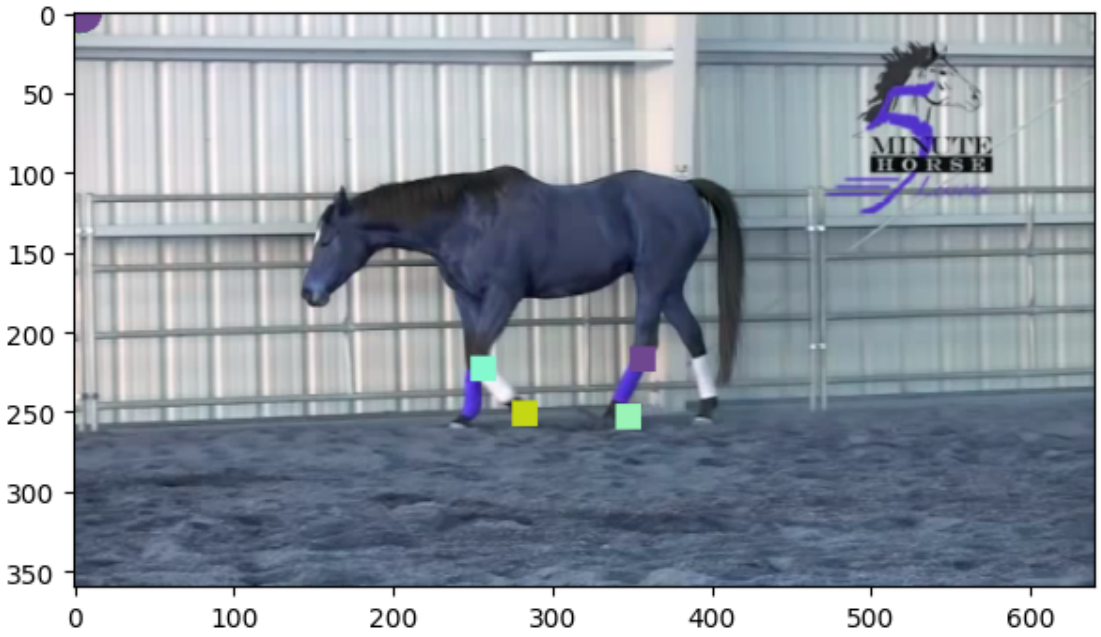


Epoch 50: Loss (3067.60498046875)

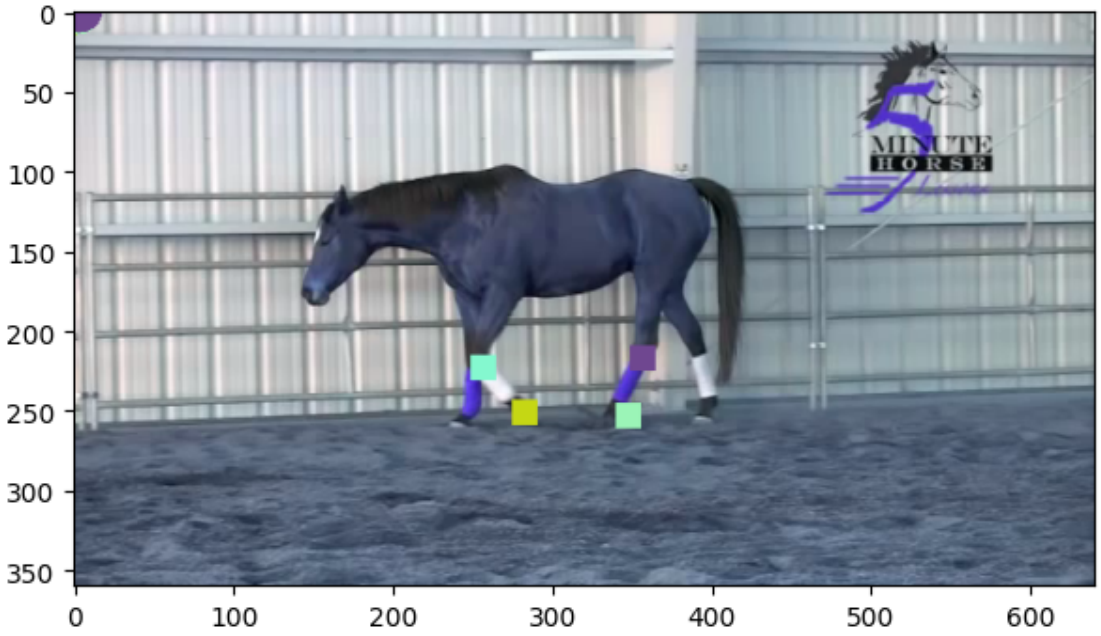
<Figure size 640x480 with 0 Axes>



```
Epoch 60: Loss (3031.44921875)
<Figure size 640x480 with 0 Axes>
```

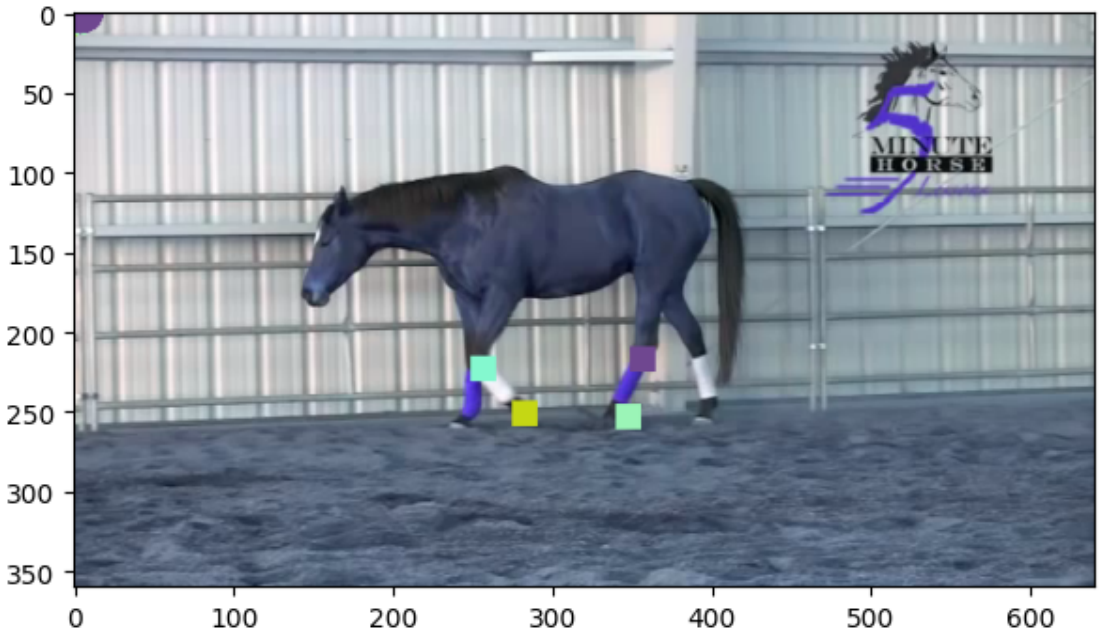



```
Epoch 70: Loss (3027.59619140625)
<Figure size 640x480 with 0 Axes>
```



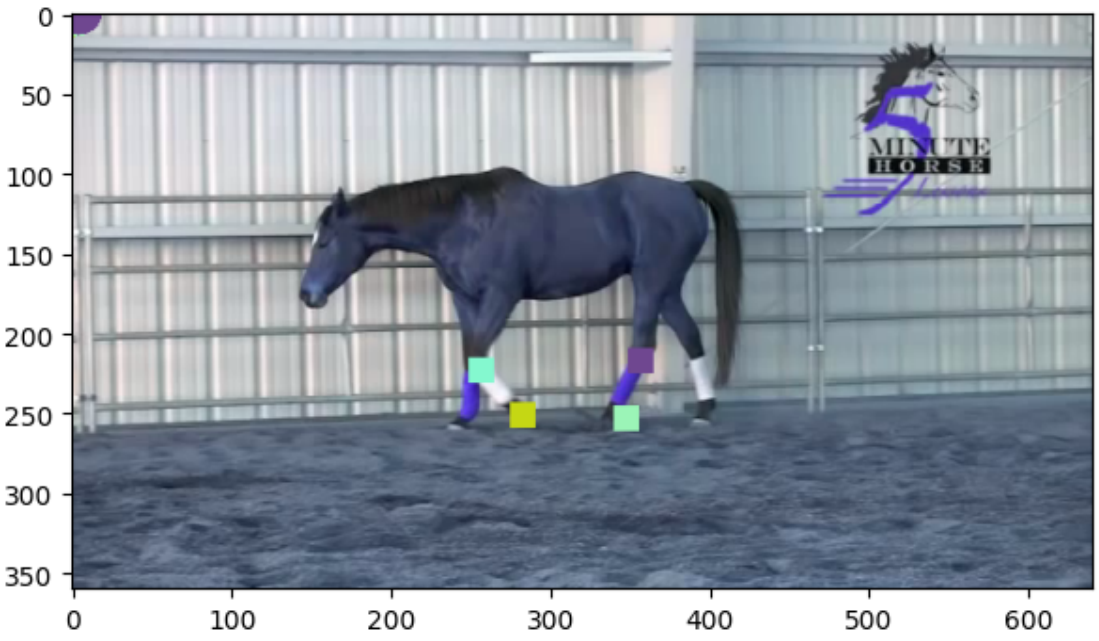
Epoch 80: Loss (3021.17724609375)

<Figure size 640x480 with 0 Axes>



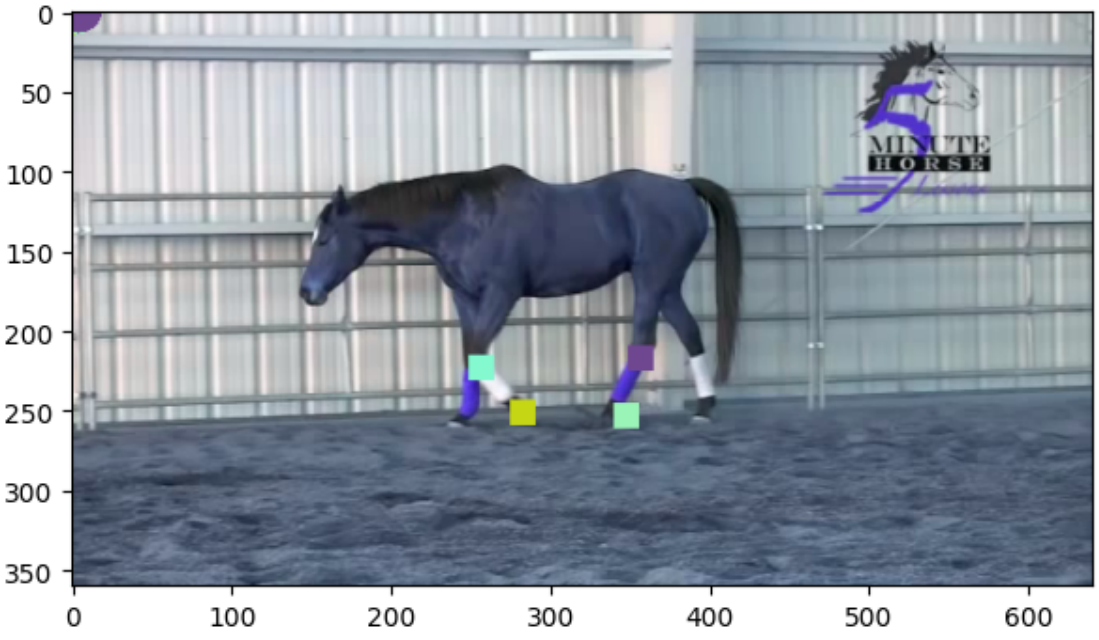
Epoch 90: Loss (3018.3134765625)

<Figure size 640x480 with 0 Axes>



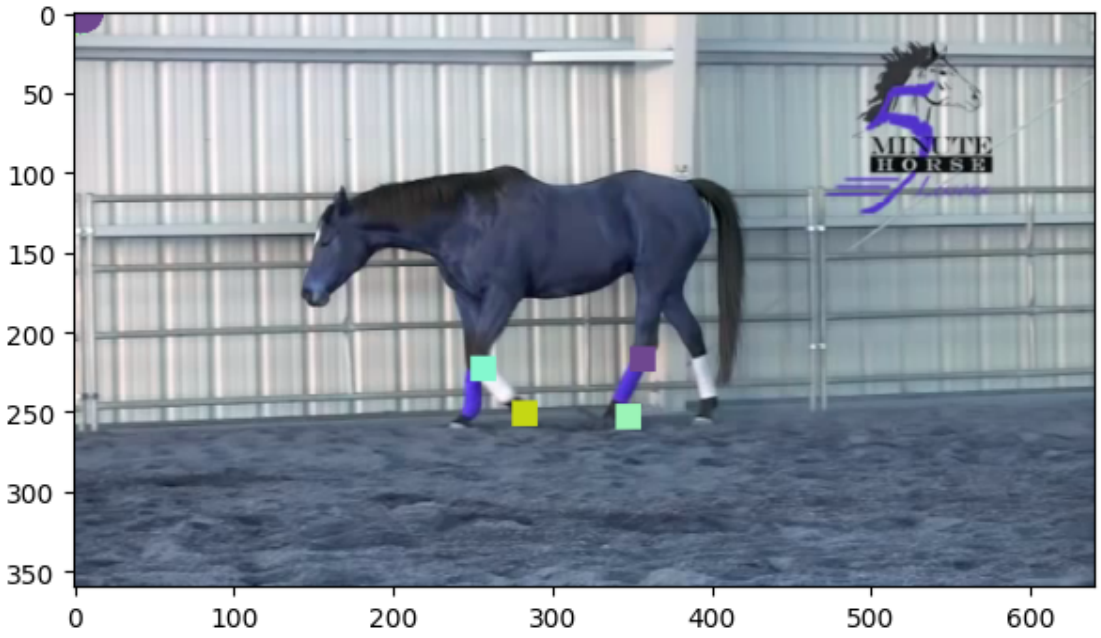
Epoch 100: Loss (3017.43505859375)

<Figure size 640x480 with 0 Axes>

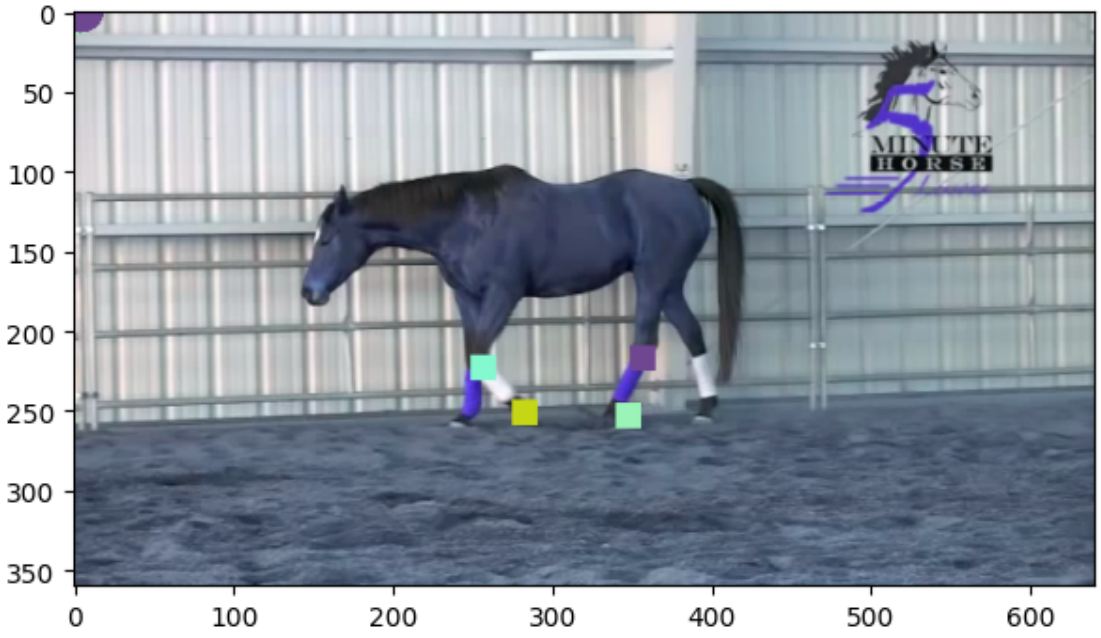


Epoch 110: Loss (3015.493408203125)

```
<Figure size 640x480 with 0 Axes>
```

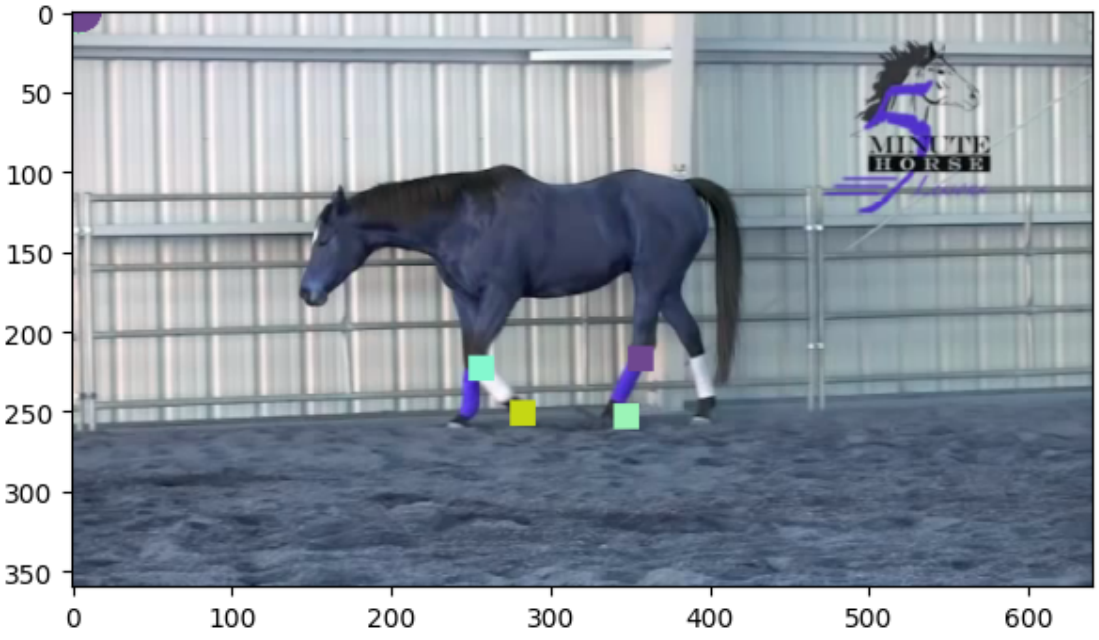



```
Epoch 120: Loss (3012.94873046875)
<Figure size 640x480 with 0 Axes>
```



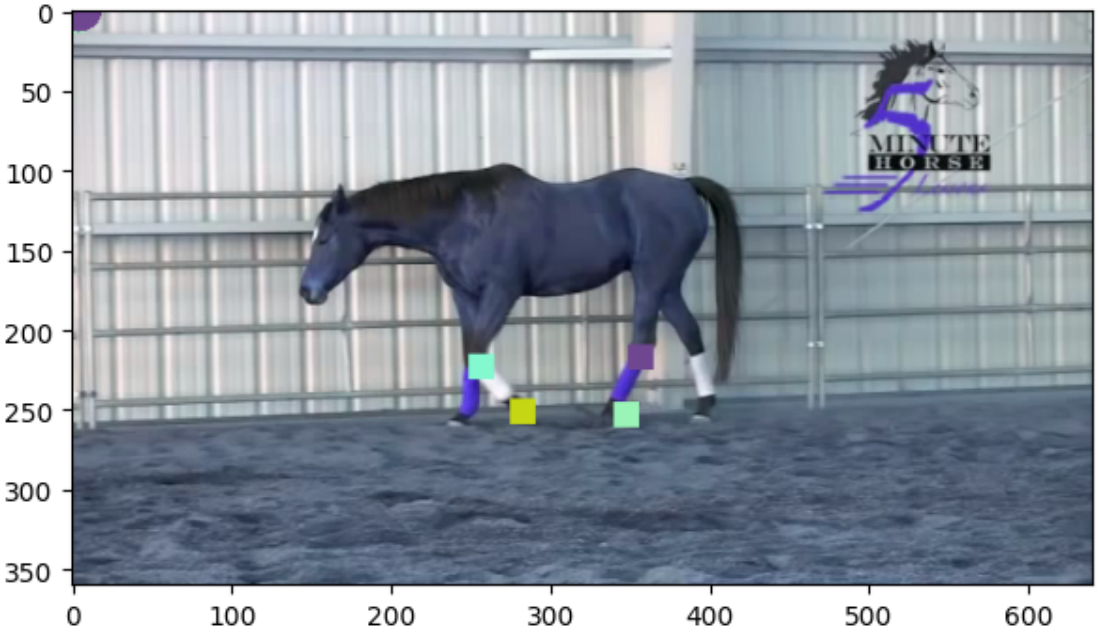
Epoch 130: Loss (3006.50830078125)

<Figure size 640x480 with 0 Axes>



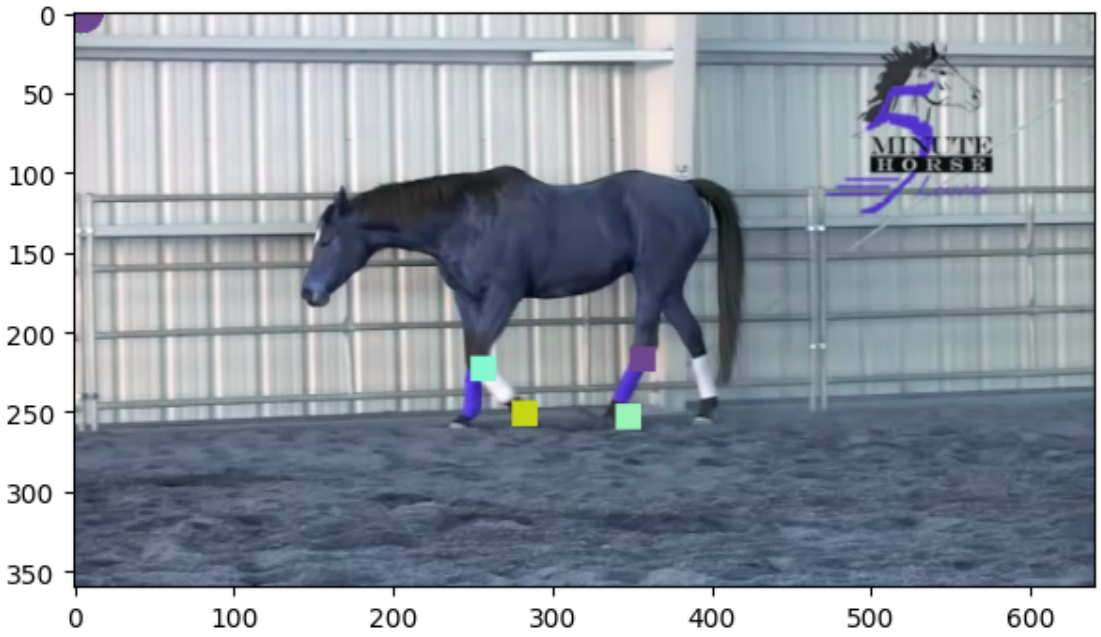
Epoch 140: Loss (3004.58837890625)

<Figure size 640x480 with 0 Axes>



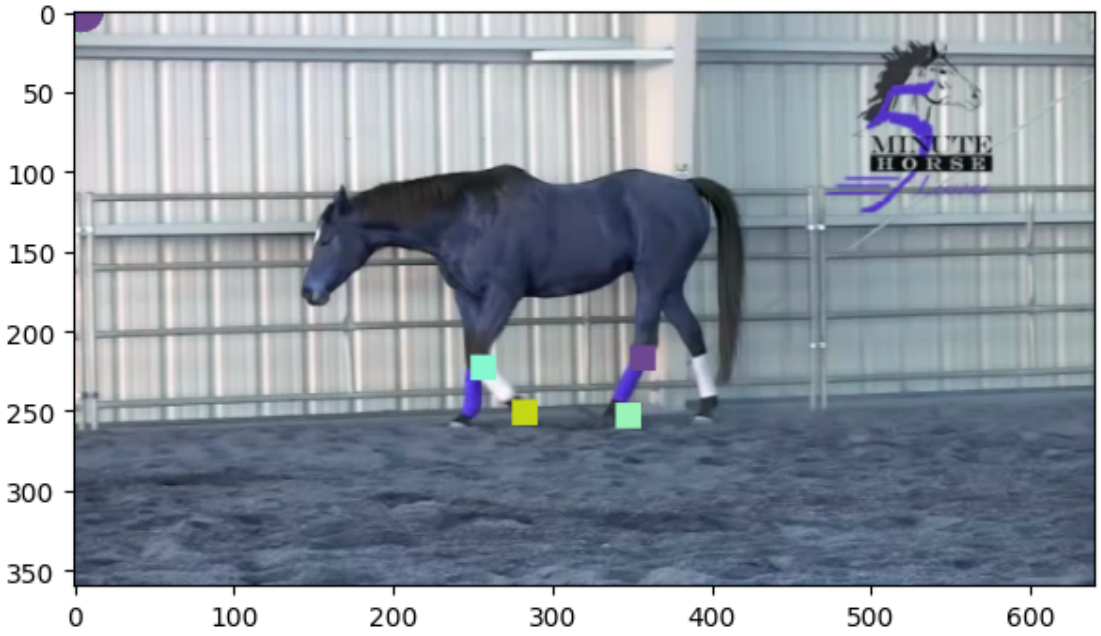
Epoch 150: Loss (3003.49755859375)

<Figure size 640x480 with 0 Axes>



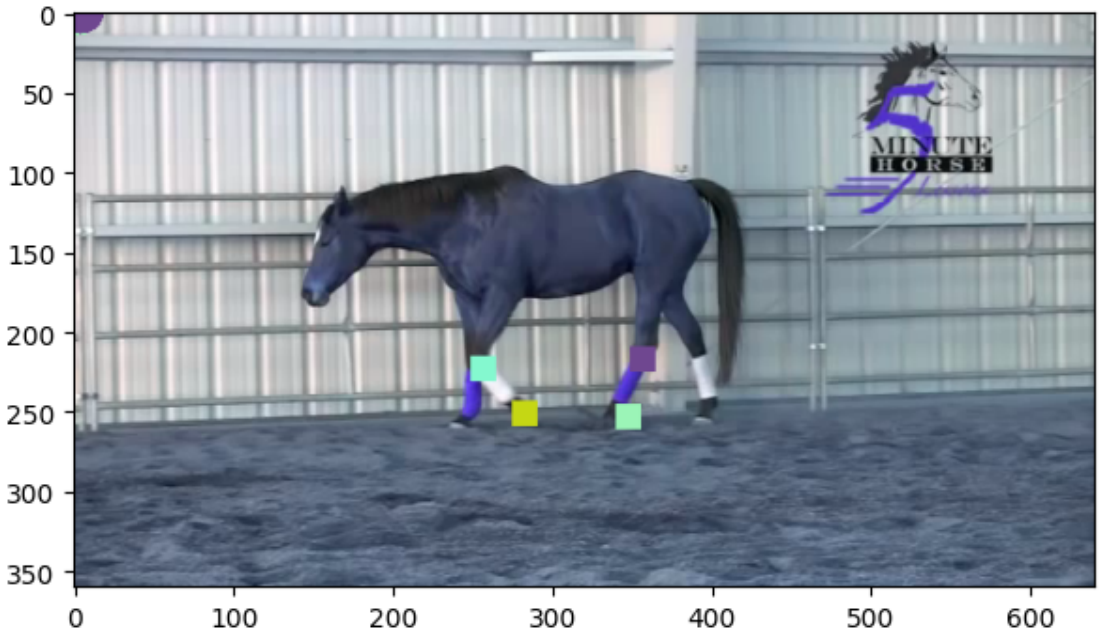
Epoch 160: Loss (3002.7490234375)

<Figure size 640x480 with 0 Axes>



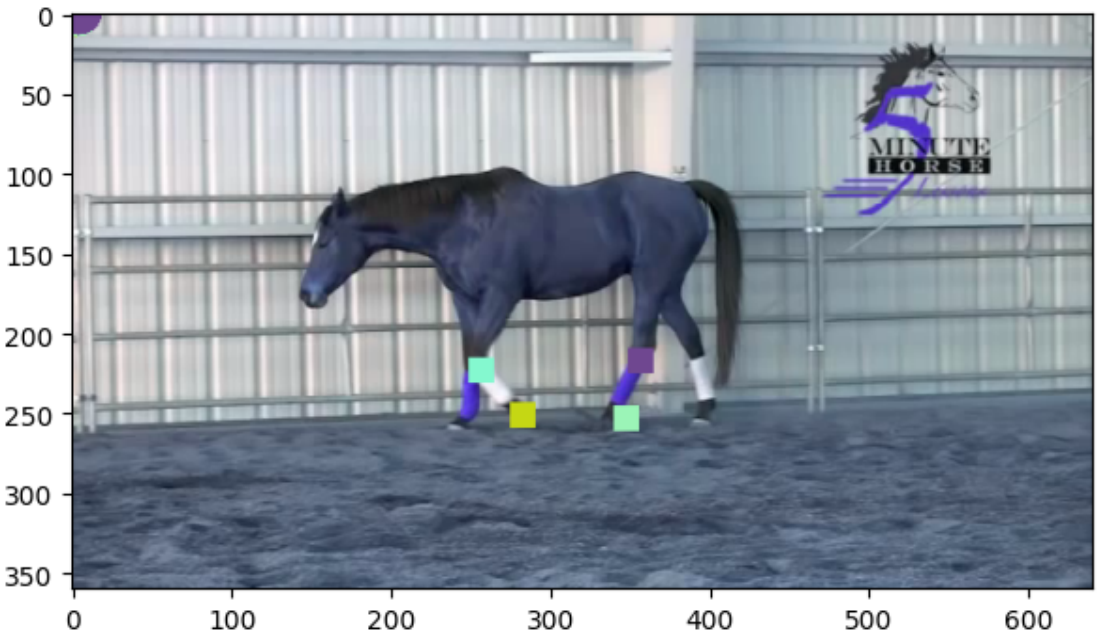
Epoch 170: Loss (3001.966796875)

<Figure size 640x480 with 0 Axes>

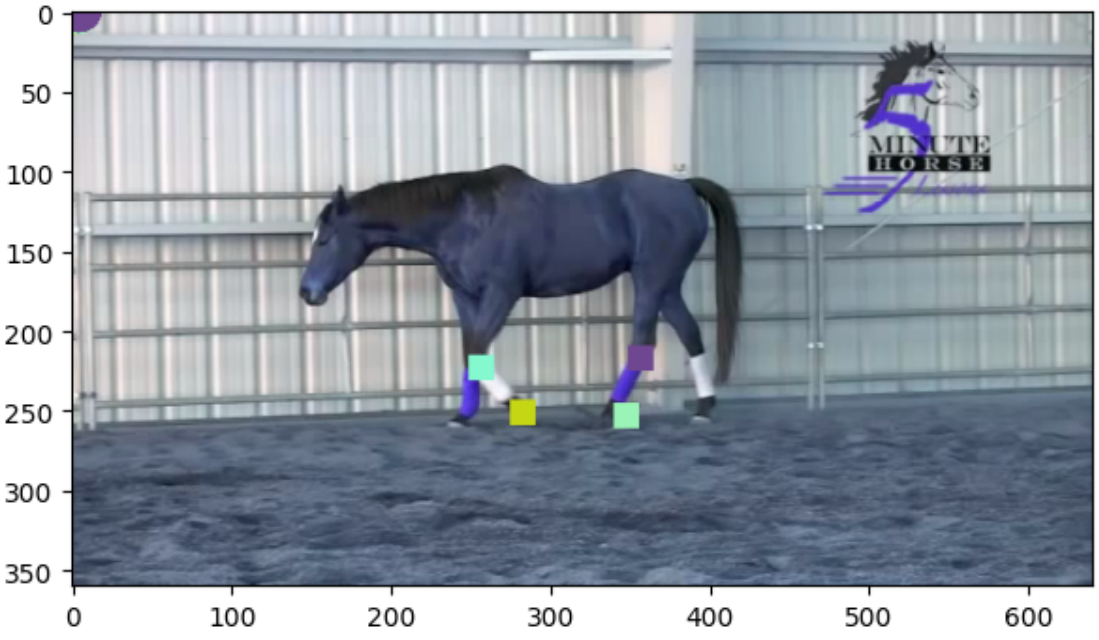


Epoch 180: Loss (3001.19775390625)

<Figure size 640x480 with 0 Axes>

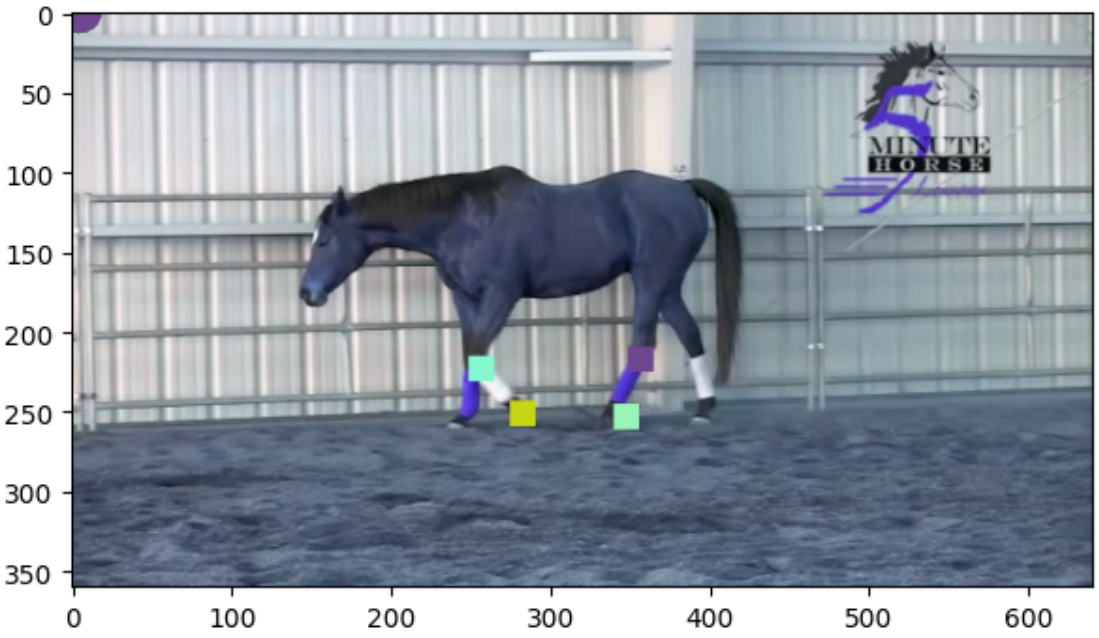



```
Epoch 190: Loss (3000.409912109375)
<Figure size 640x480 with 0 Axes>
```



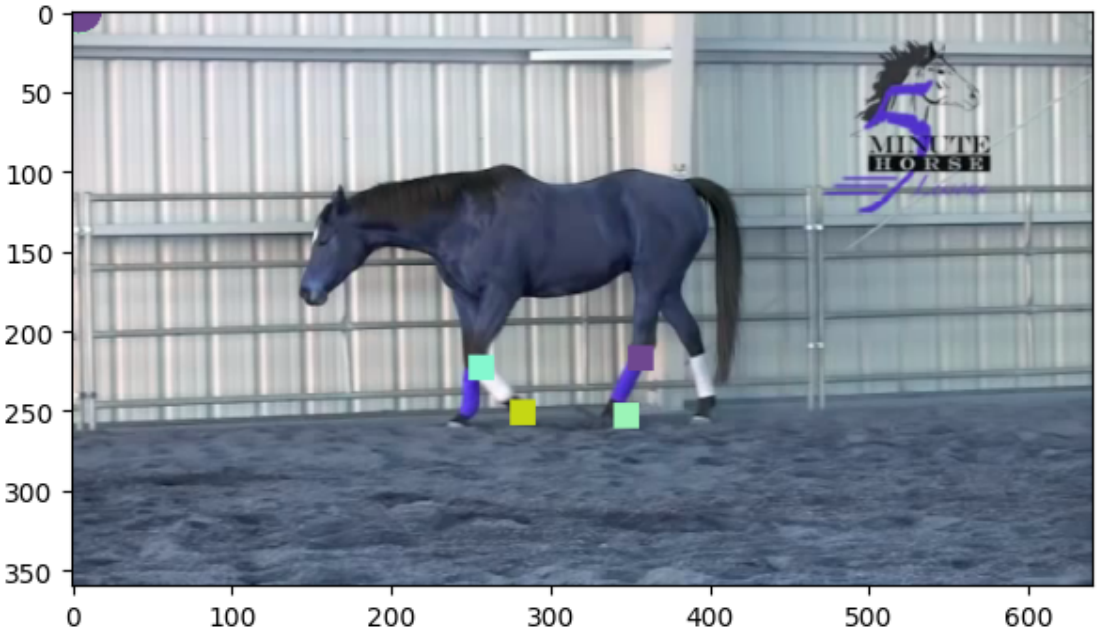

```
Epoch 200: Loss (2999.6025390625)
```

<Figure size 640x480 with 0 Axes>

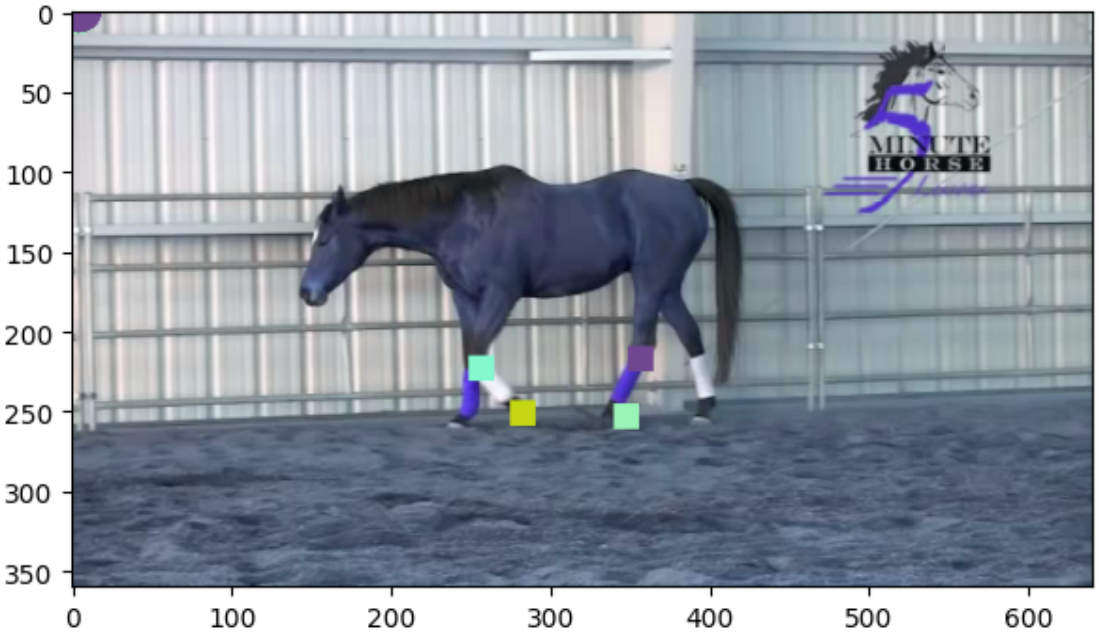


Epoch 210: Loss (2998.763671875)

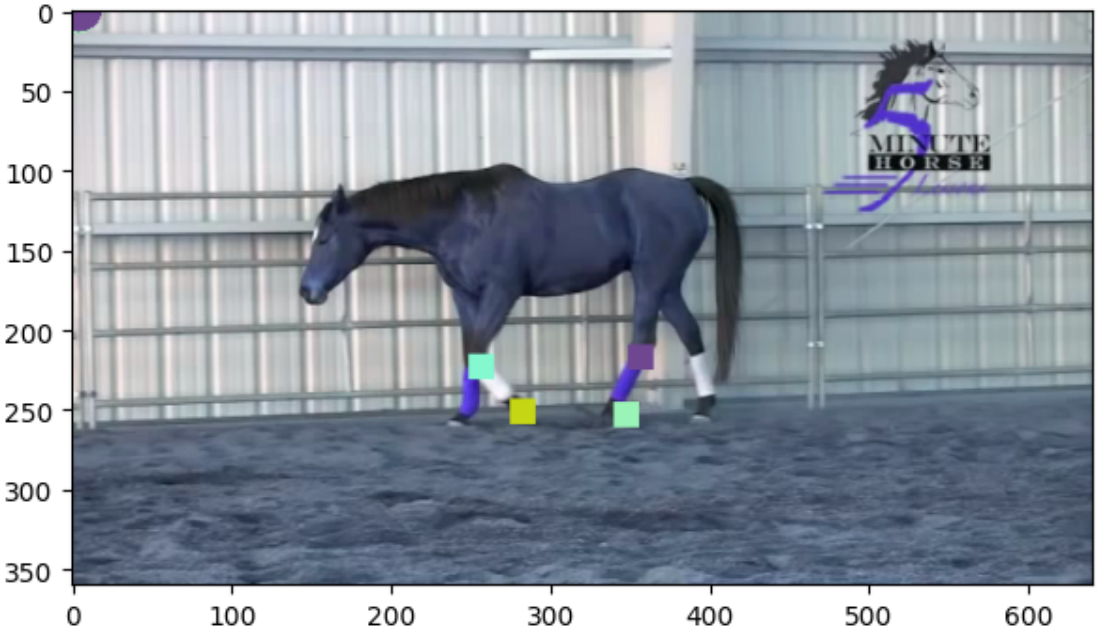
<Figure size 640x480 with 0 Axes>



```
Epoch 220: Loss (2997.8740234375)
<Figure size 640x480 with 0 Axes>
```

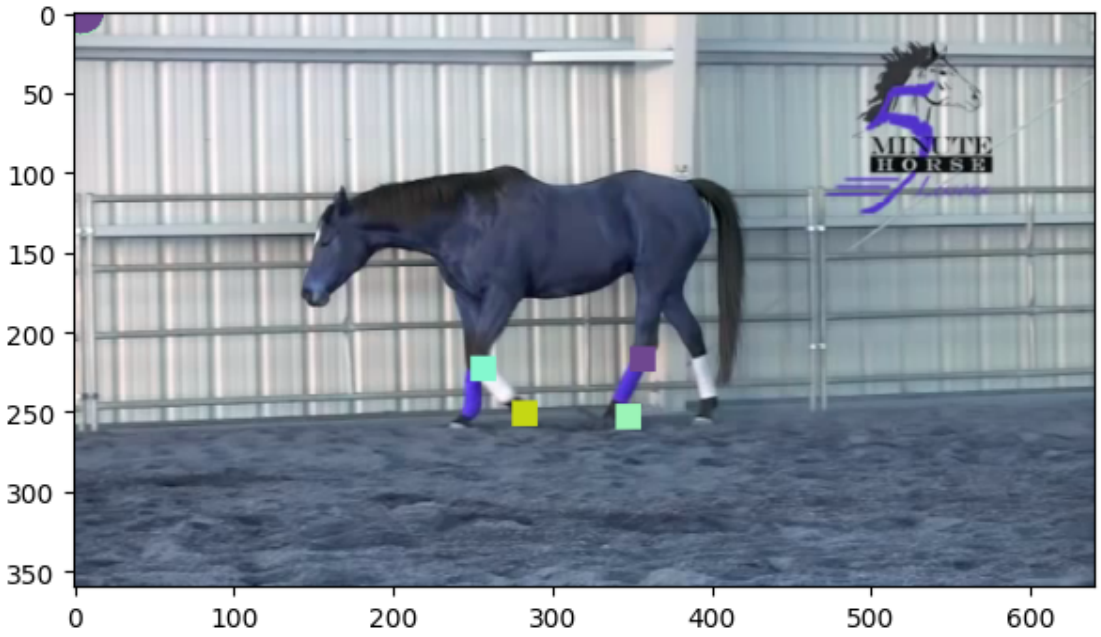



```
Epoch 230: Loss (2996.899658203125)
<Figure size 640x480 with 0 Axes>
```



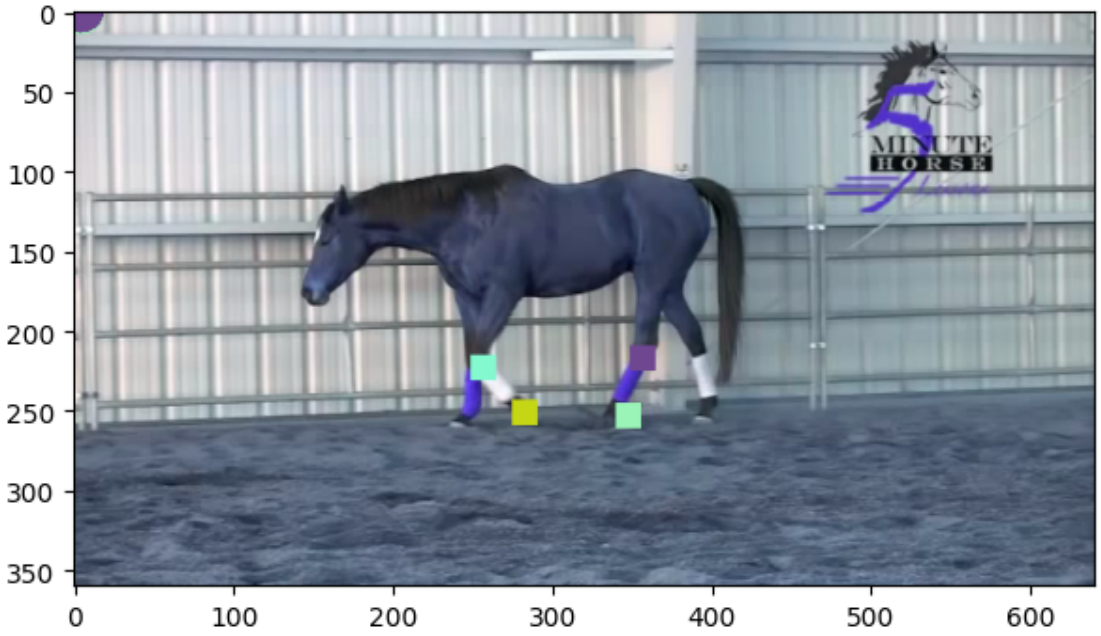
Epoch 240: Loss (2995.77490234375)

<Figure size 640x480 with 0 Axes>



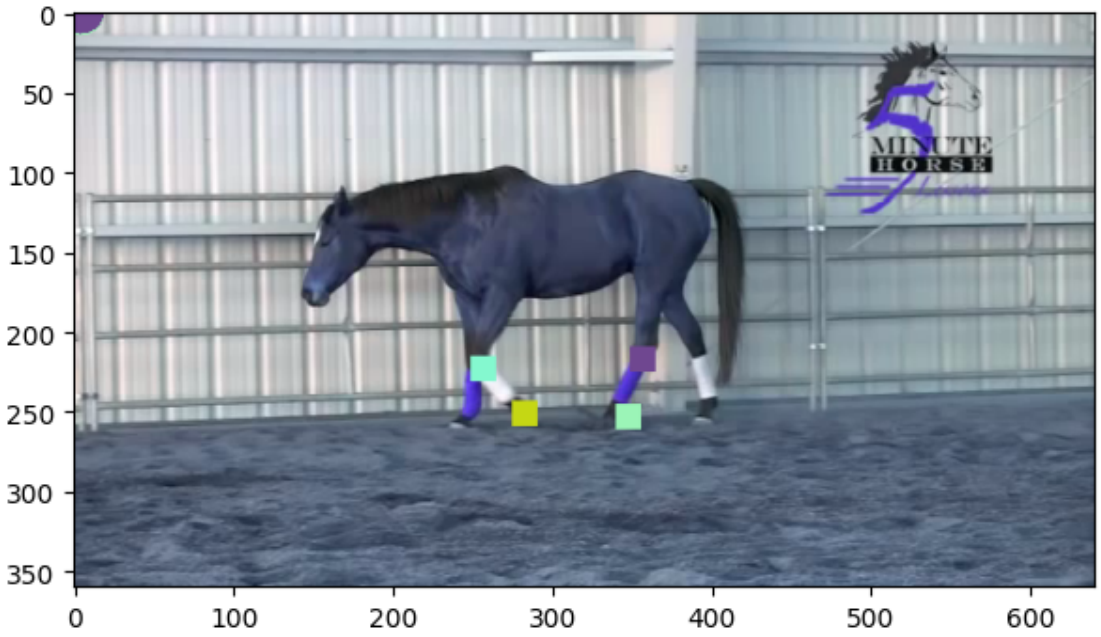
Epoch 250: Loss (2994.340576171875)

<Figure size 640x480 with 0 Axes>



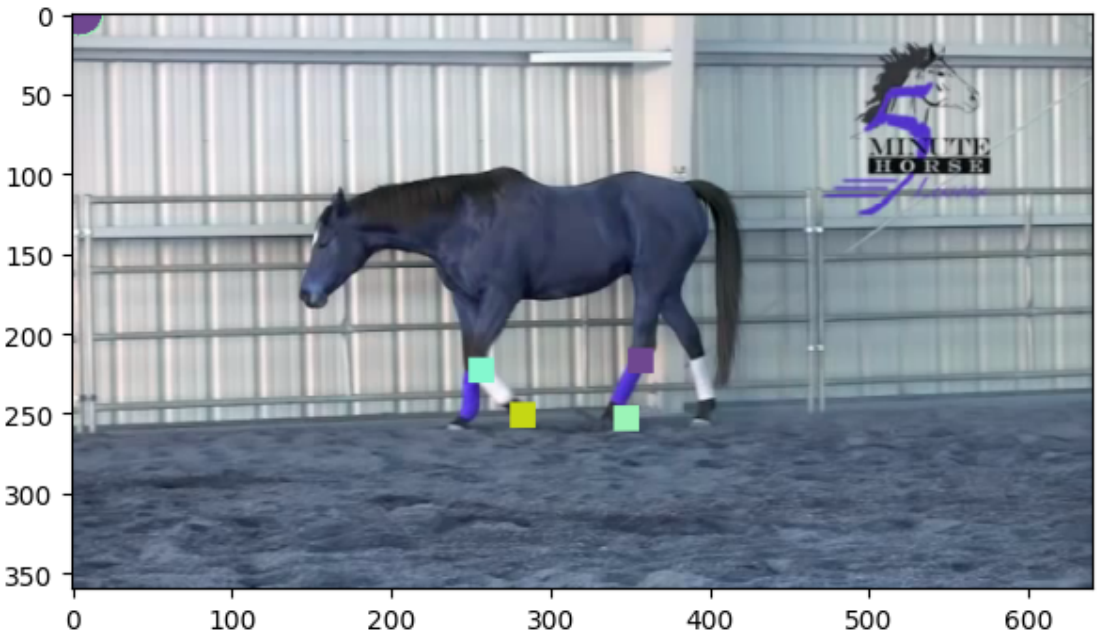
Epoch 260: Loss (2992.078125)

<Figure size 640x480 with 0 Axes>



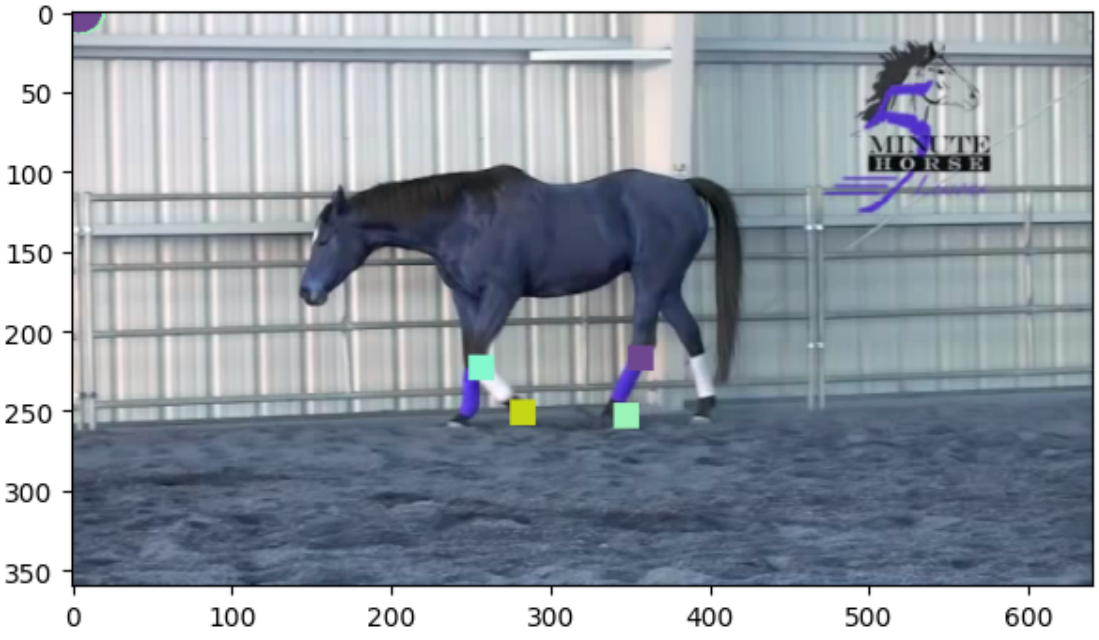
Epoch 270: Loss (2986.124267578125)

<Figure size 640x480 with 0 Axes>



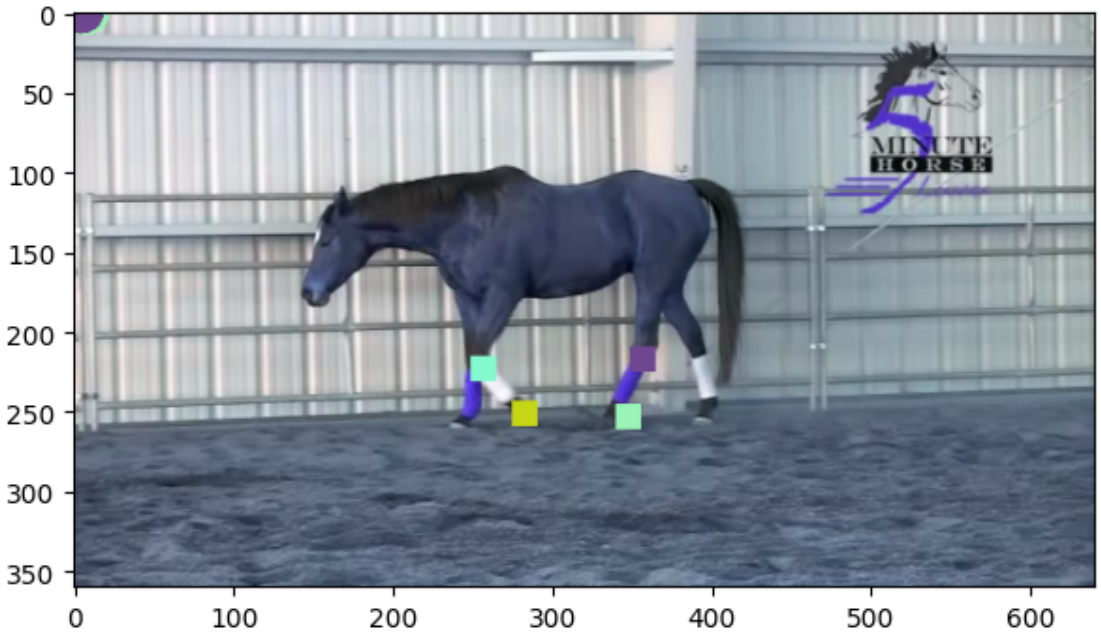
Epoch 280: Loss (2983.49853515625)

<Figure size 640x480 with 0 Axes>



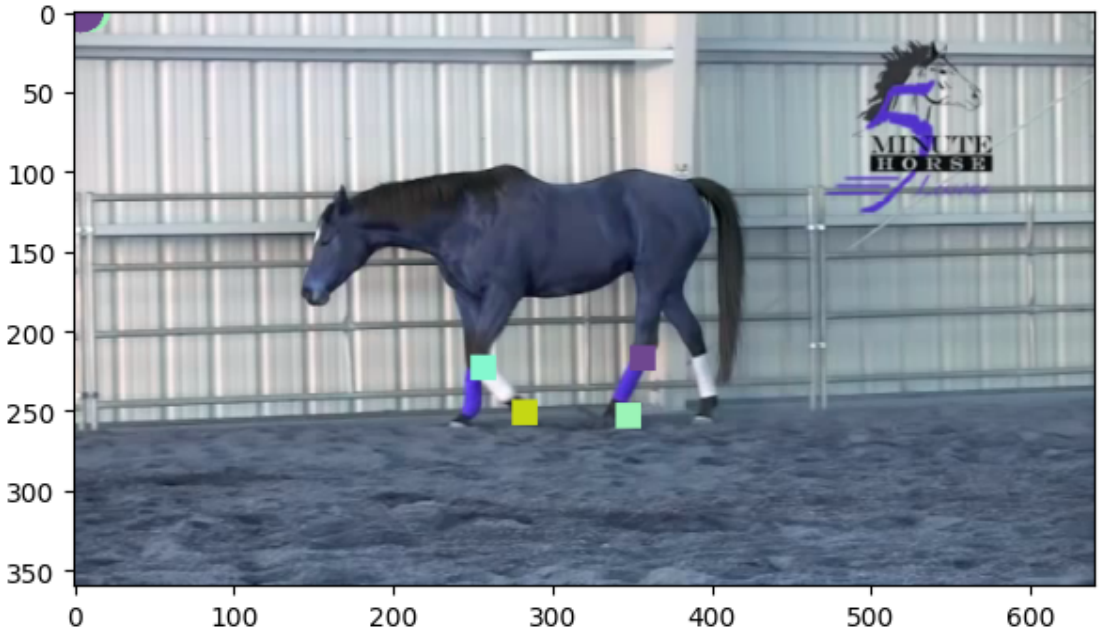
Epoch 290: Loss (2981.84130859375)

<Figure size 640x480 with 0 Axes>



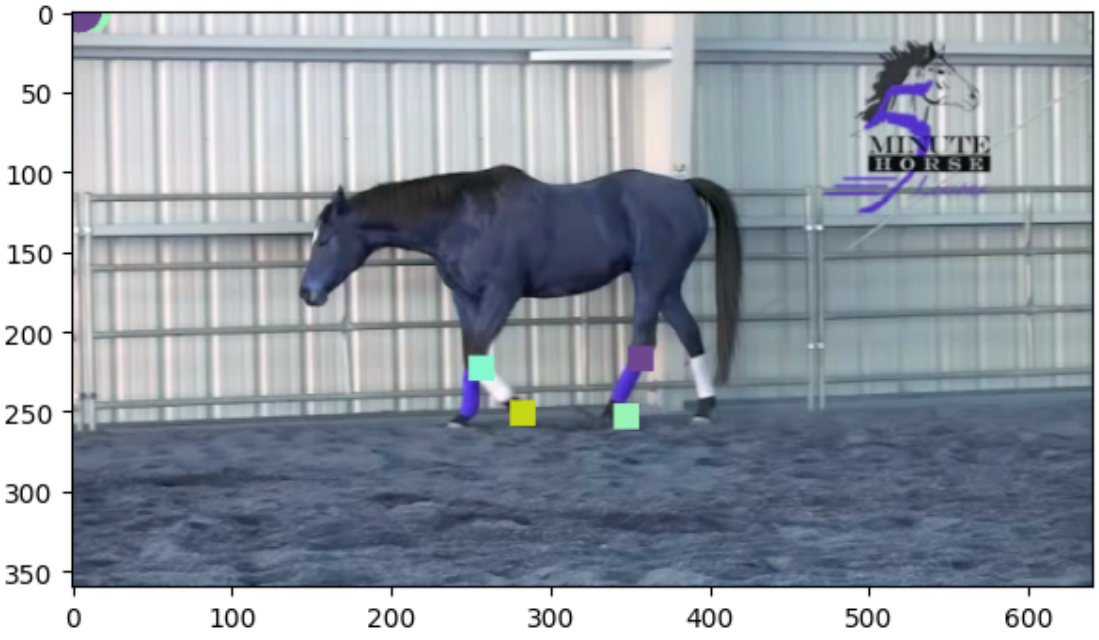
Epoch 300: Loss (2980.345947265625)

<Figure size 640x480 with 0 Axes>



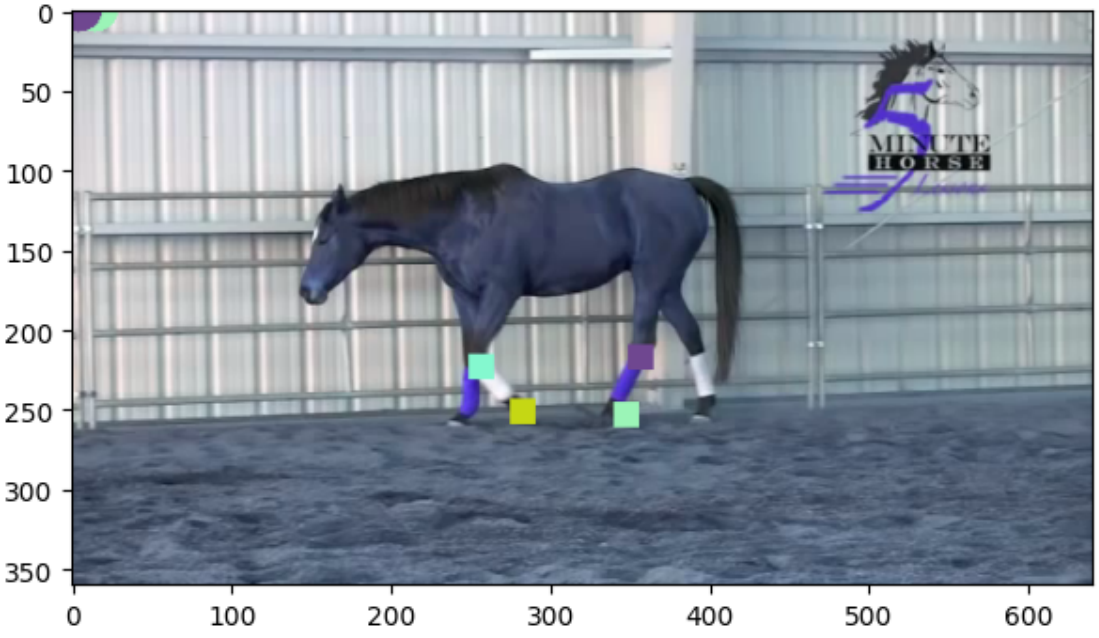
Epoch 310: Loss (2978.42578125)

<Figure size 640x480 with 0 Axes>

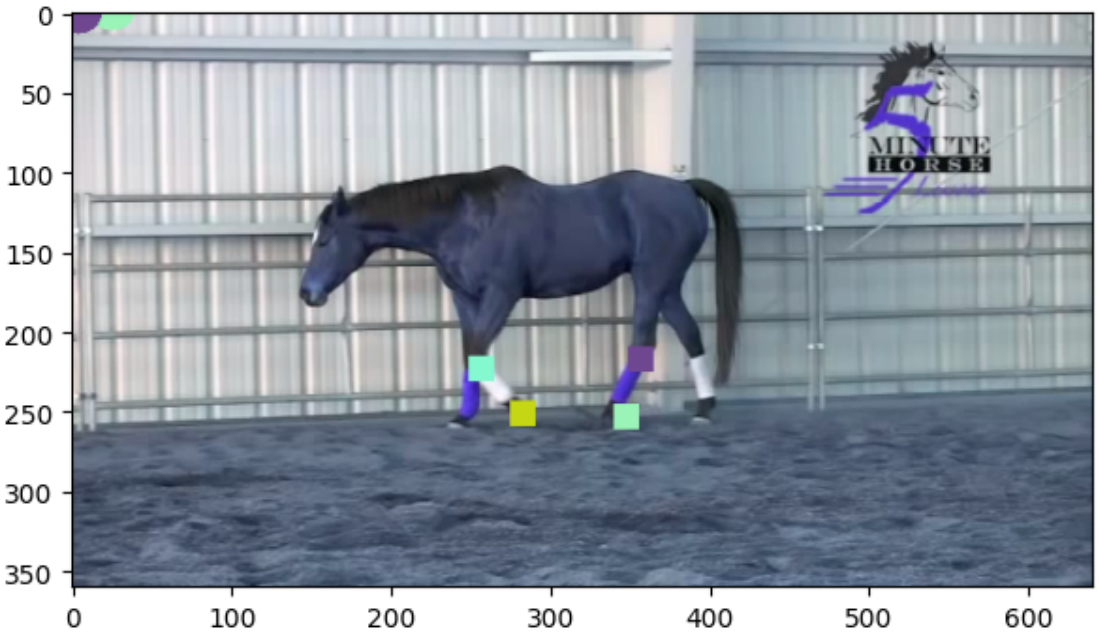


Epoch 320: Loss (2975.74462890625)

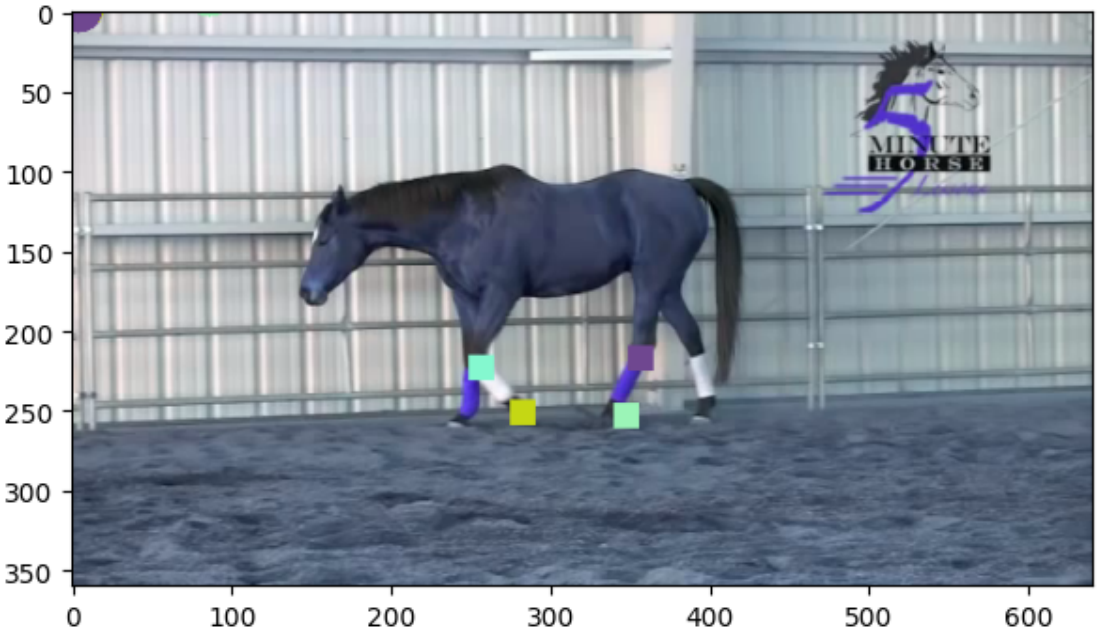
<Figure size 640x480 with 0 Axes>



```
Epoch 330: Loss (2970.496337890625)
<Figure size 640x480 with 0 Axes>
```

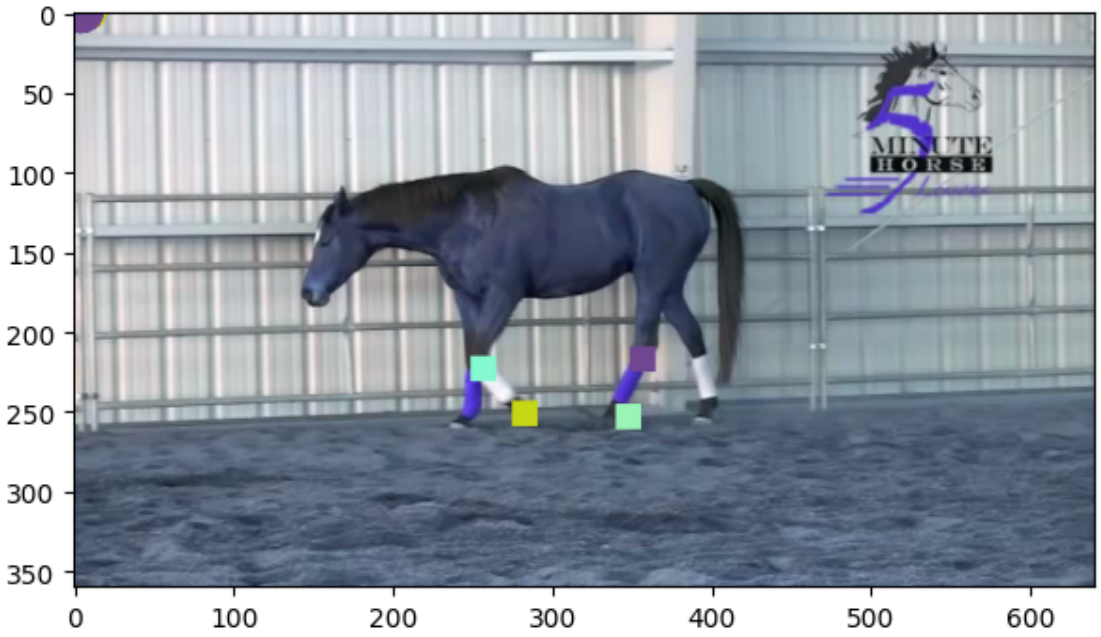



```
Epoch 340: Loss (2946.71240234375)
<Figure size 640x480 with 0 Axes>
```



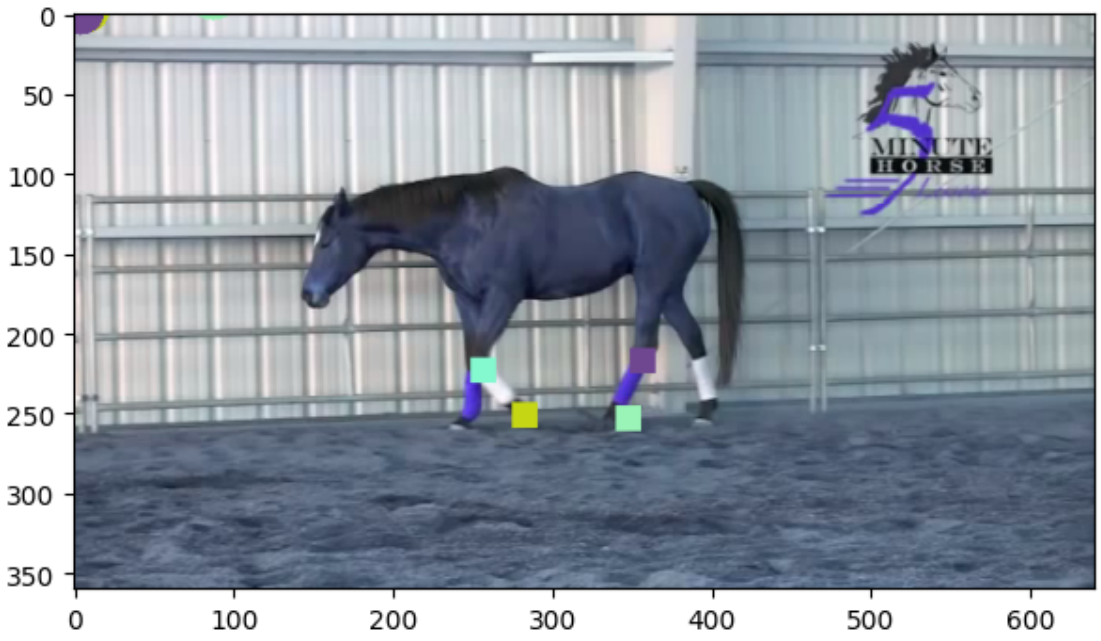
Epoch 350: Loss (2932.77587890625)

<Figure size 640x480 with 0 Axes>



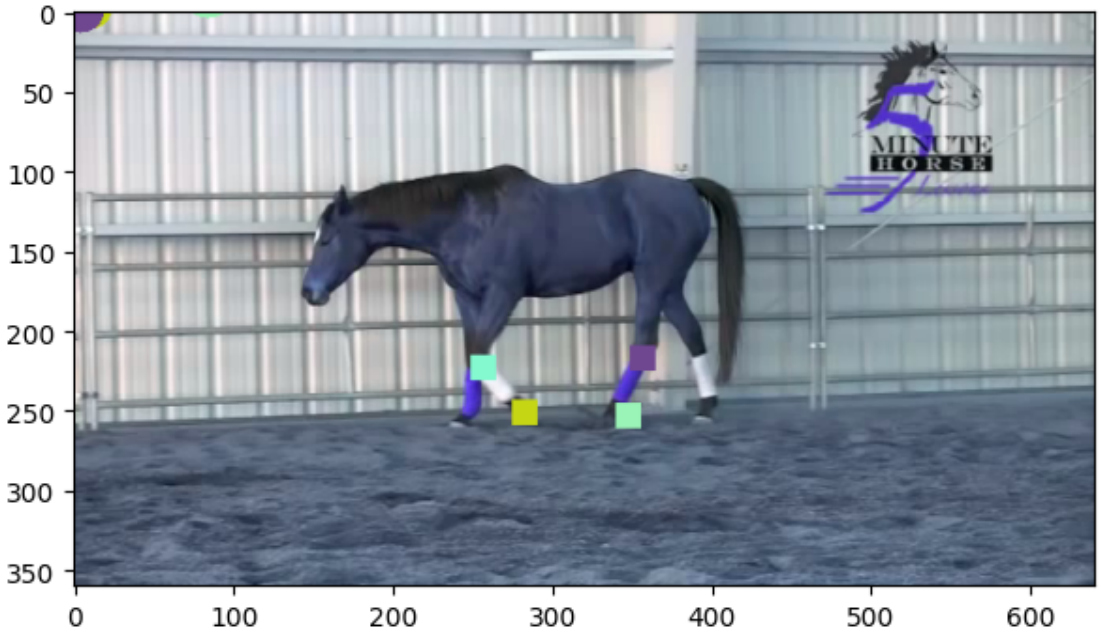
Epoch 360: Loss (2944.196533203125)

<Figure size 640x480 with 0 Axes>



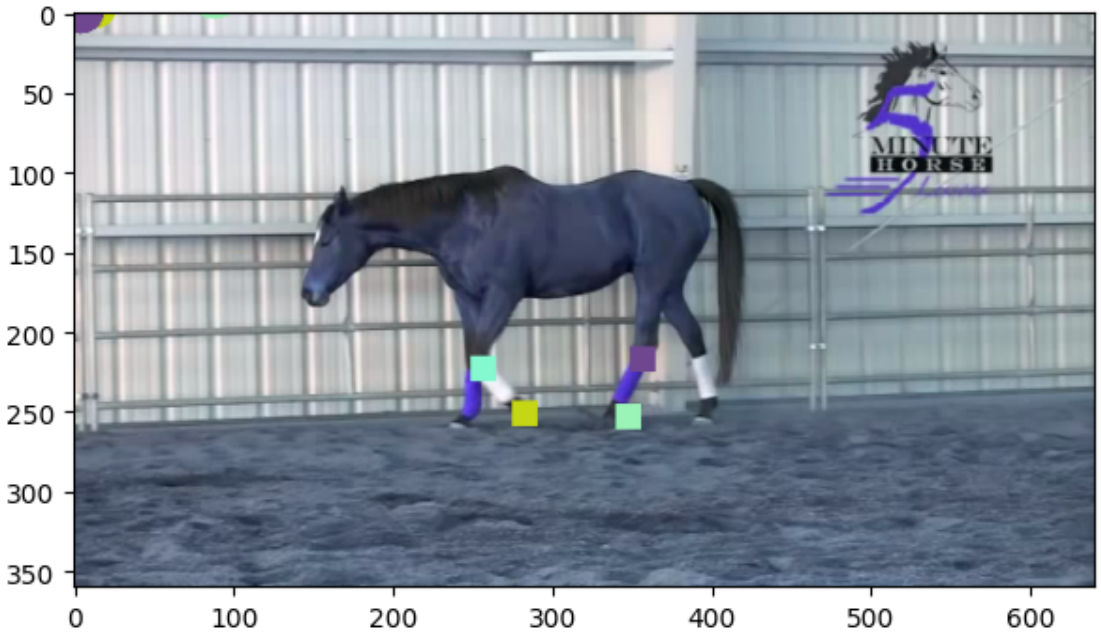
Epoch 370: Loss (2944.0361328125)

<Figure size 640x480 with 0 Axes>



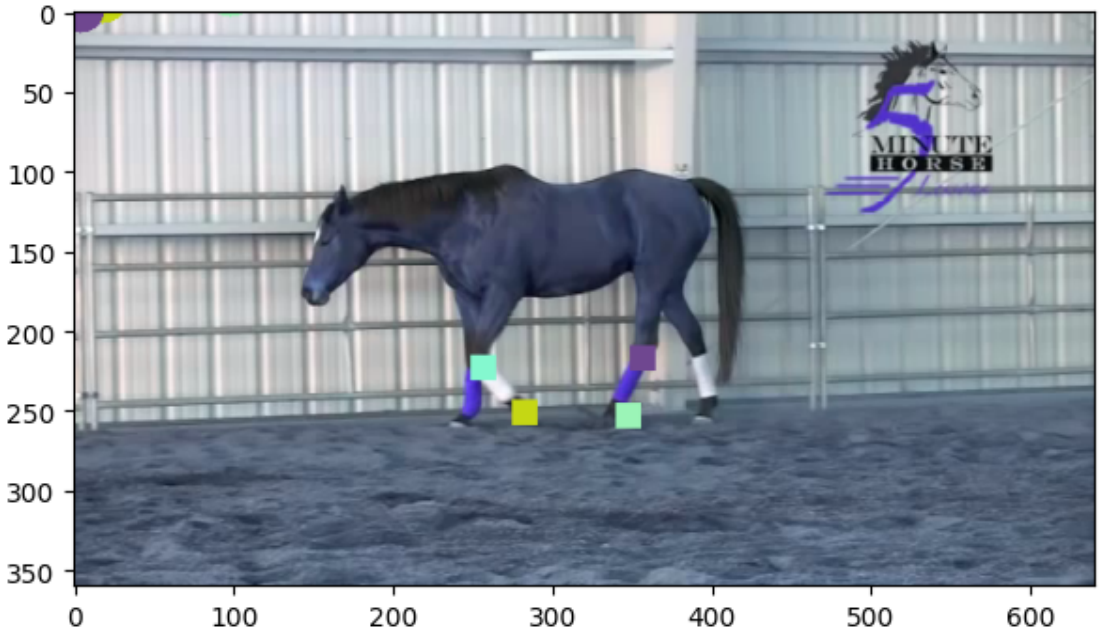
Epoch 380: Loss (2940.863037109375)

<Figure size 640x480 with 0 Axes>



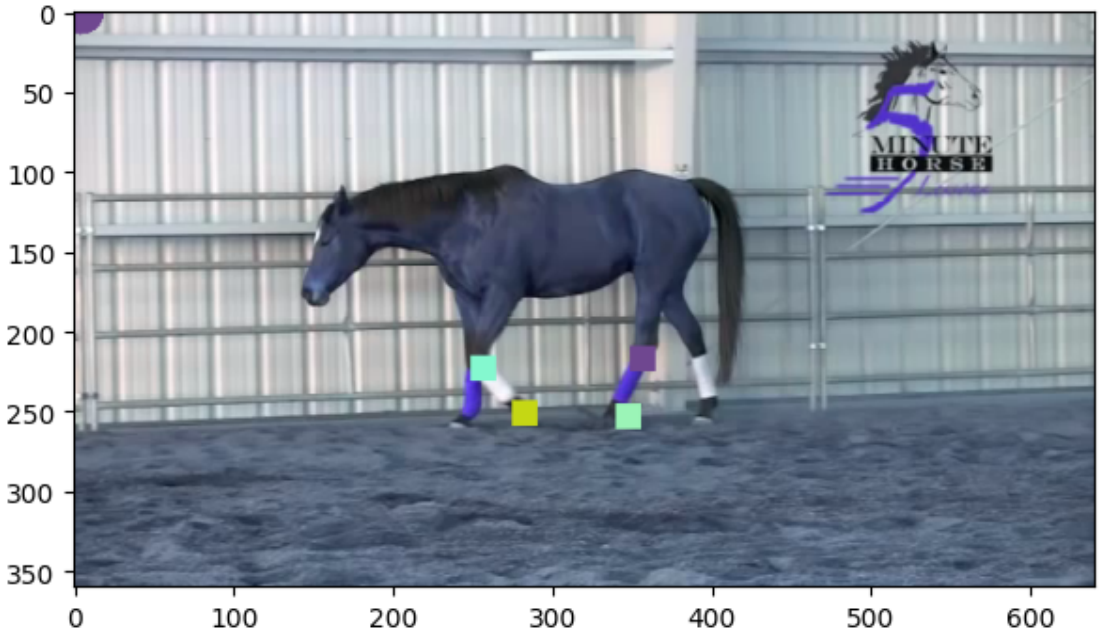
Epoch 390: Loss (2935.4736328125)

<Figure size 640x480 with 0 Axes>



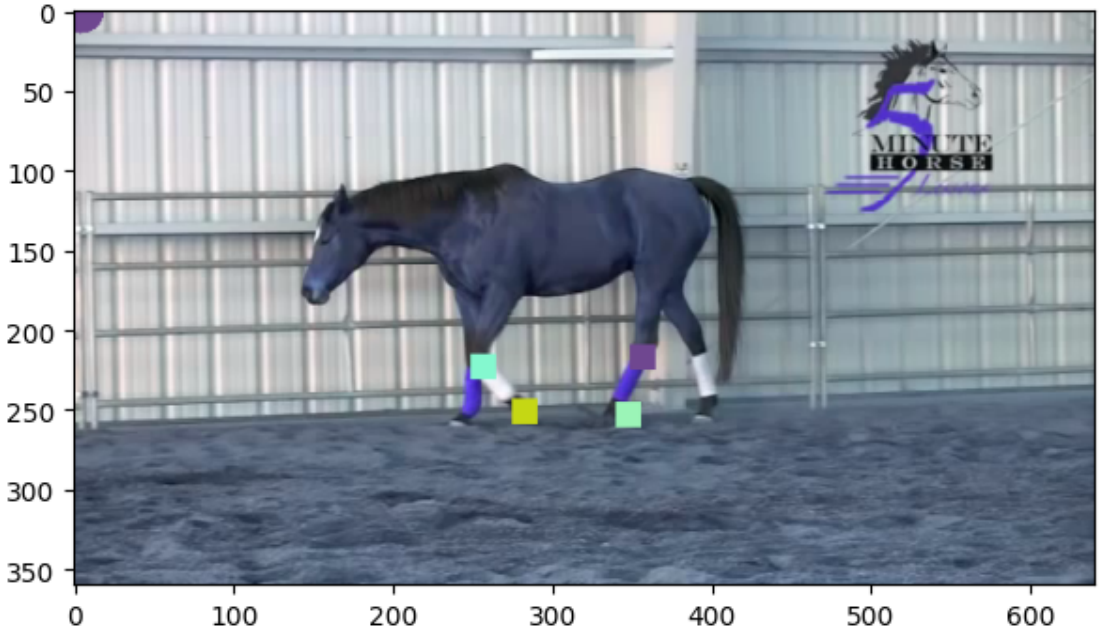
Epoch 400: Loss (2925.7431640625)

<Figure size 640x480 with 0 Axes>

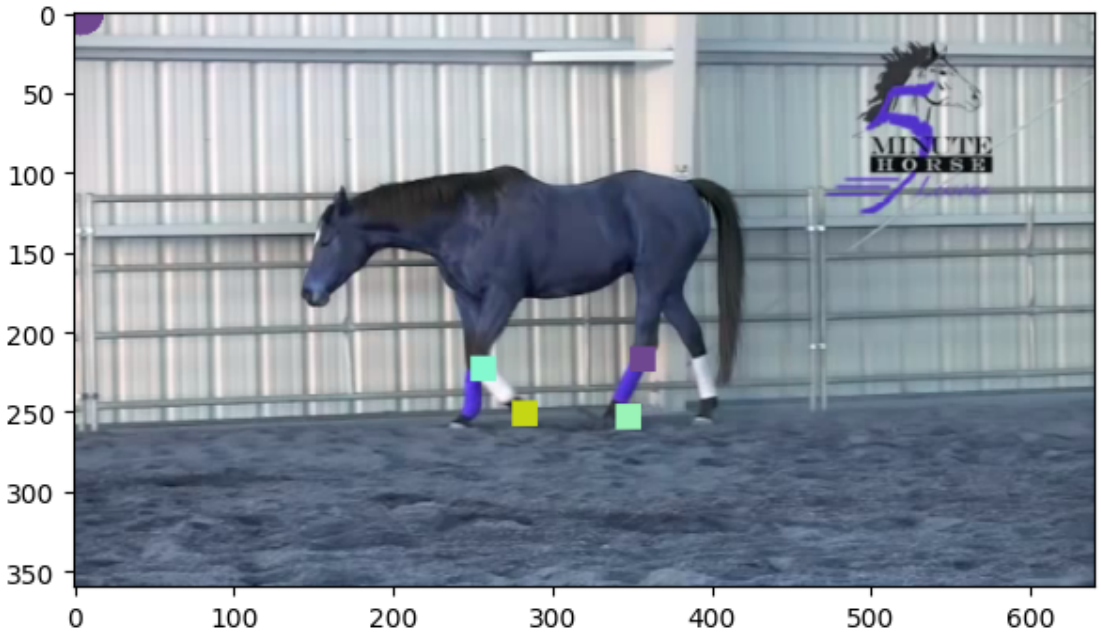


Epoch 410: Loss (2909.00634765625)

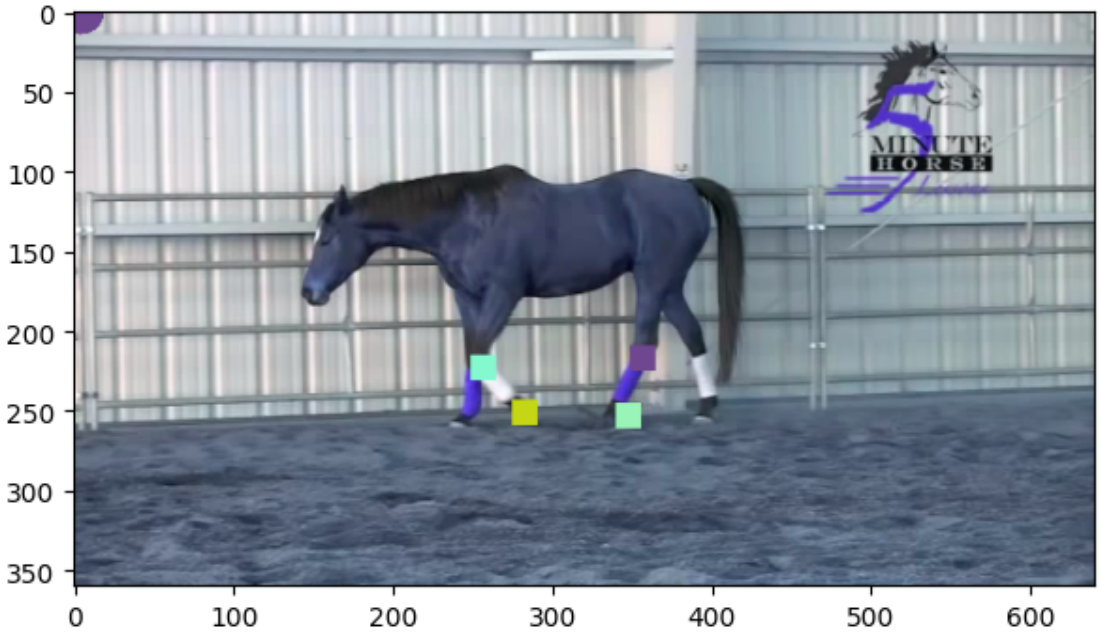
<Figure size 640x480 with 0 Axes>



```
Epoch 420: Loss (2890.247314453125)
<Figure size 640x480 with 0 Axes>
```

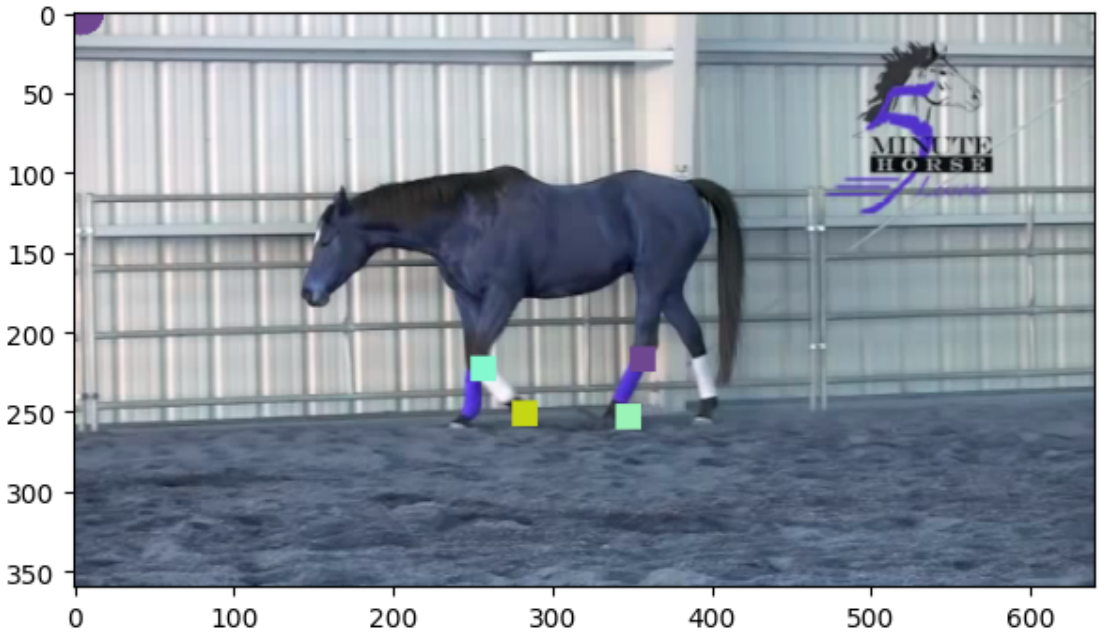



```
Epoch 430: Loss (2882.388671875)
<Figure size 640x480 with 0 Axes>
```



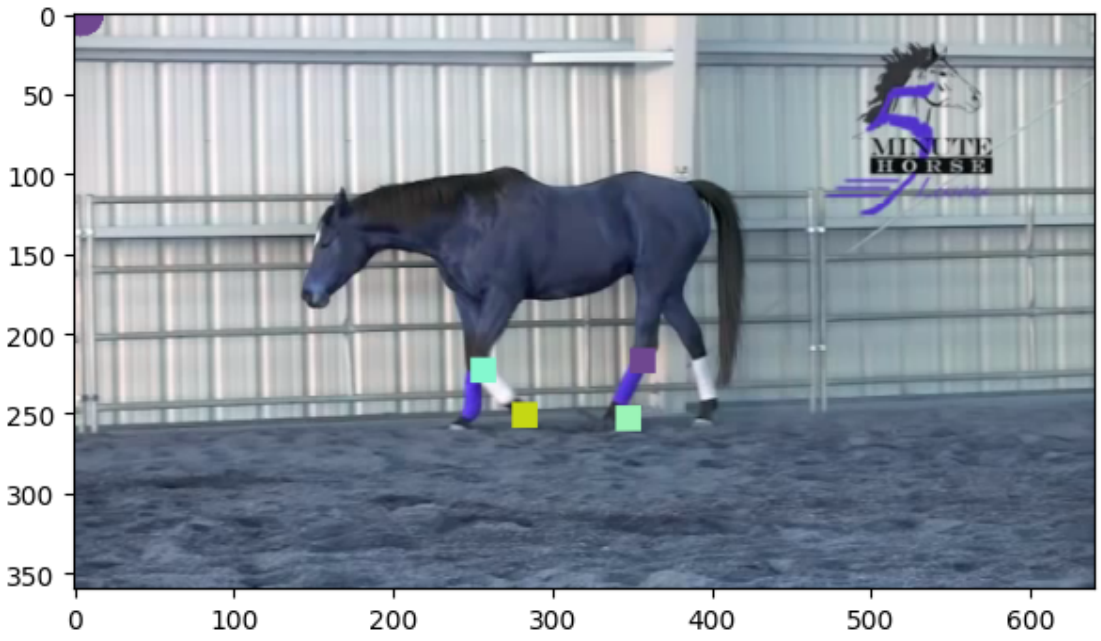
Epoch 440: Loss (2880.350830078125)

<Figure size 640x480 with 0 Axes>



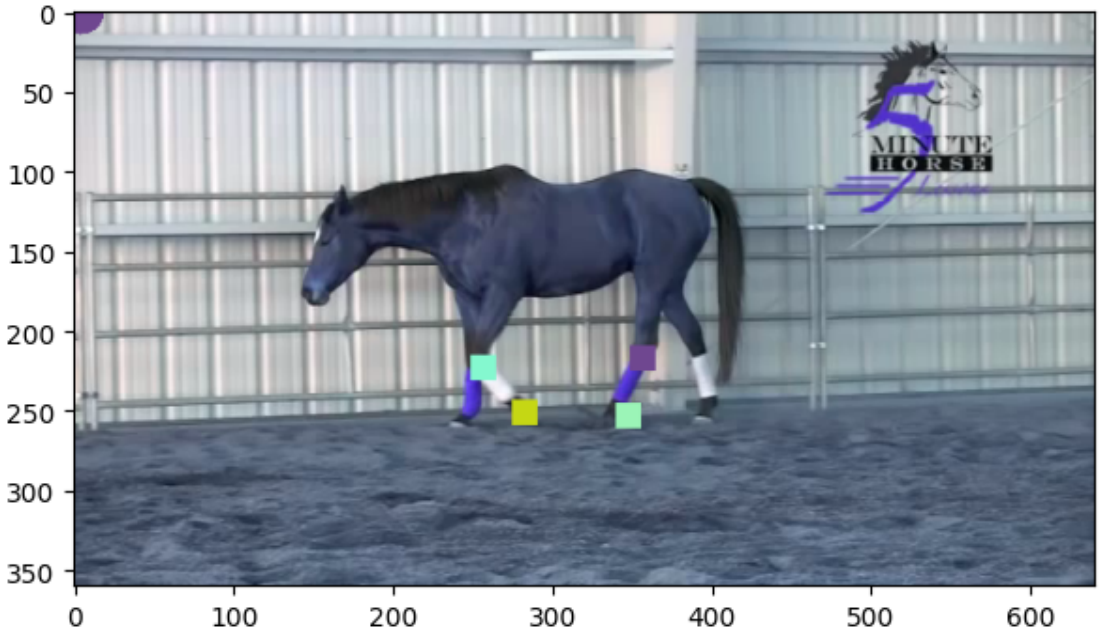
Epoch 450: Loss (2876.513427734375)

<Figure size 640x480 with 0 Axes>



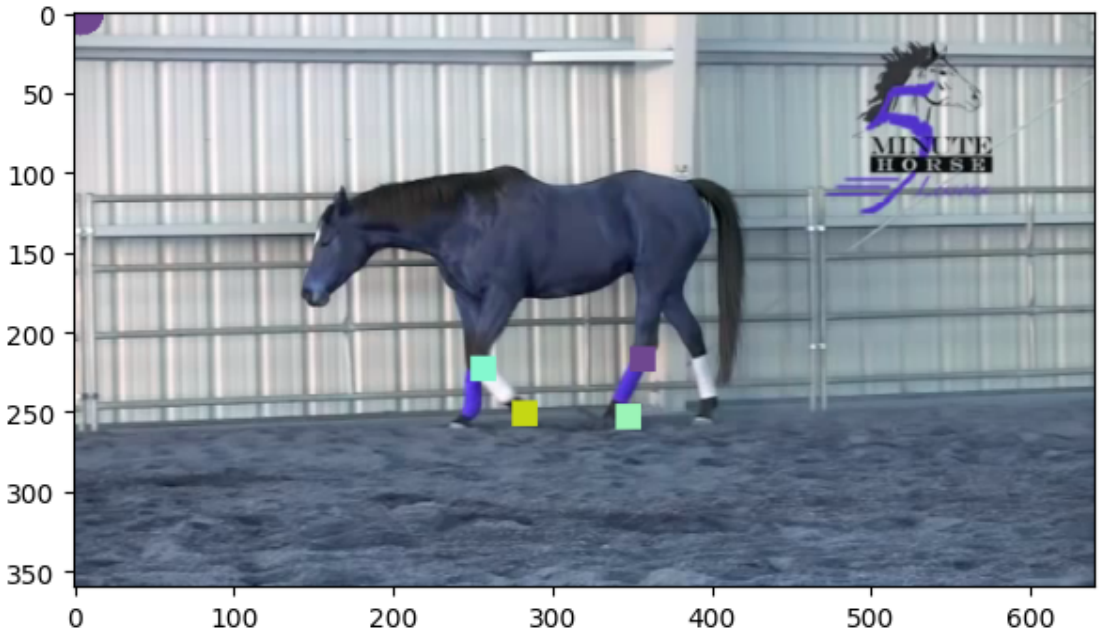
Epoch 460: Loss (2871.65673828125)

<Figure size 640x480 with 0 Axes>

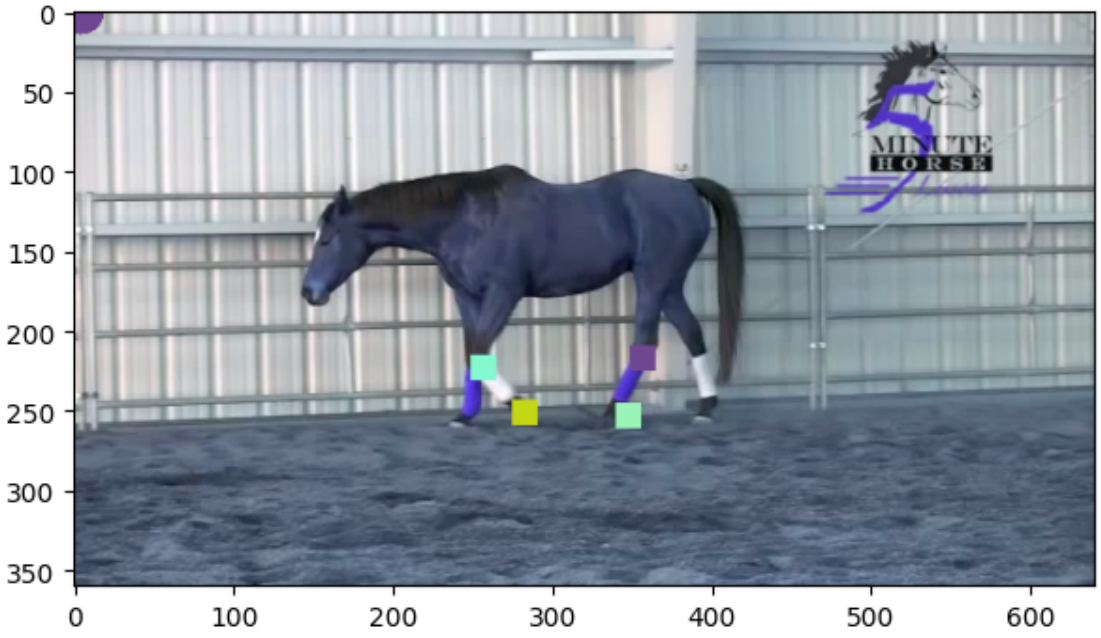


Epoch 470: Loss (2859.81884765625)

```
<Figure size 640x480 with 0 Axes>
```

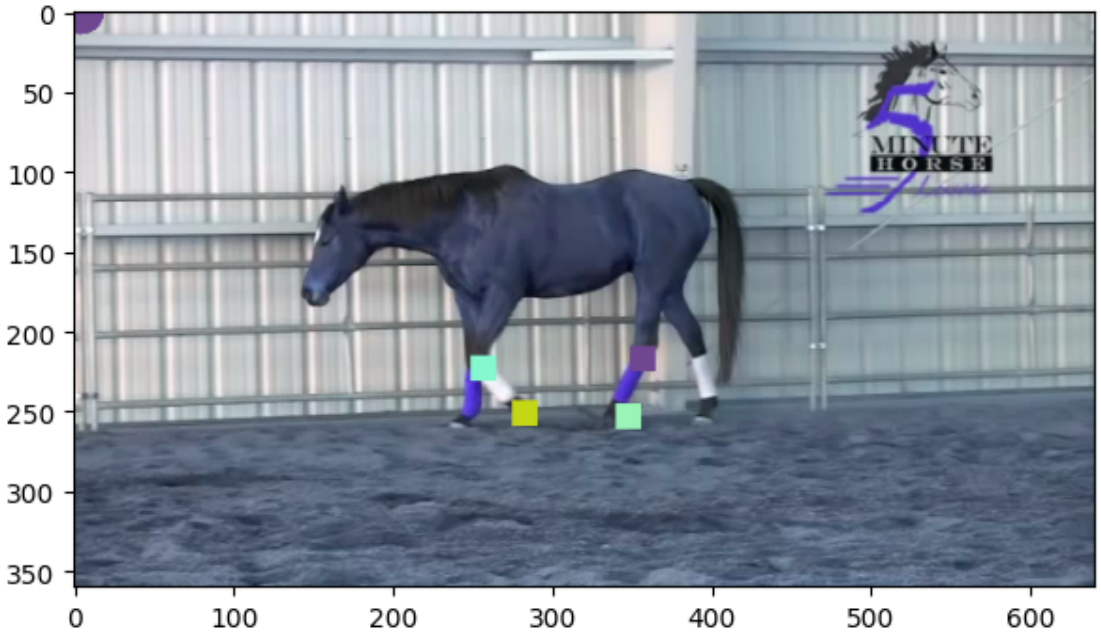



```
Epoch 480: Loss (2802.18798828125)
<Figure size 640x480 with 0 Axes>
```



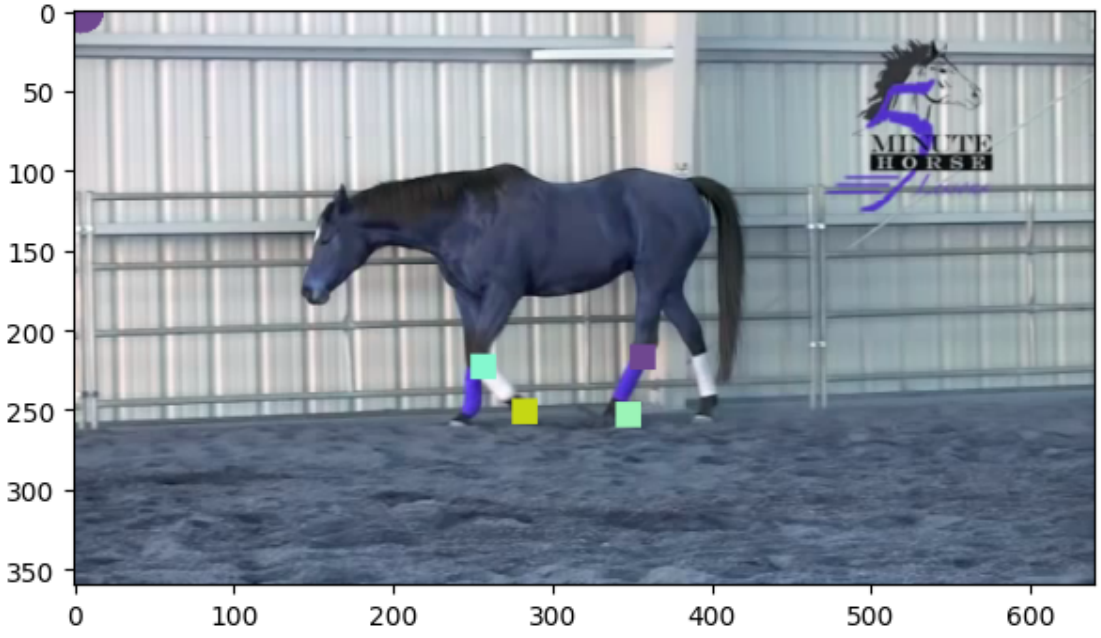
Epoch 490: Loss (2796.9755859375)

<Figure size 640x480 with 0 Axes>



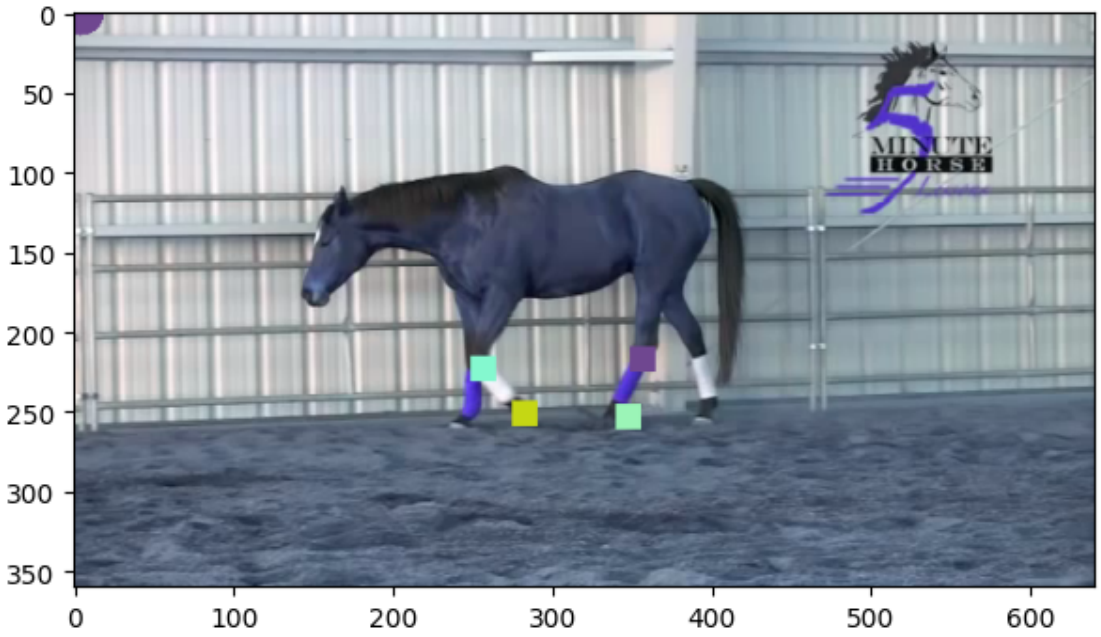
Epoch 500: Loss (2794.90869140625)

<Figure size 640x480 with 0 Axes>



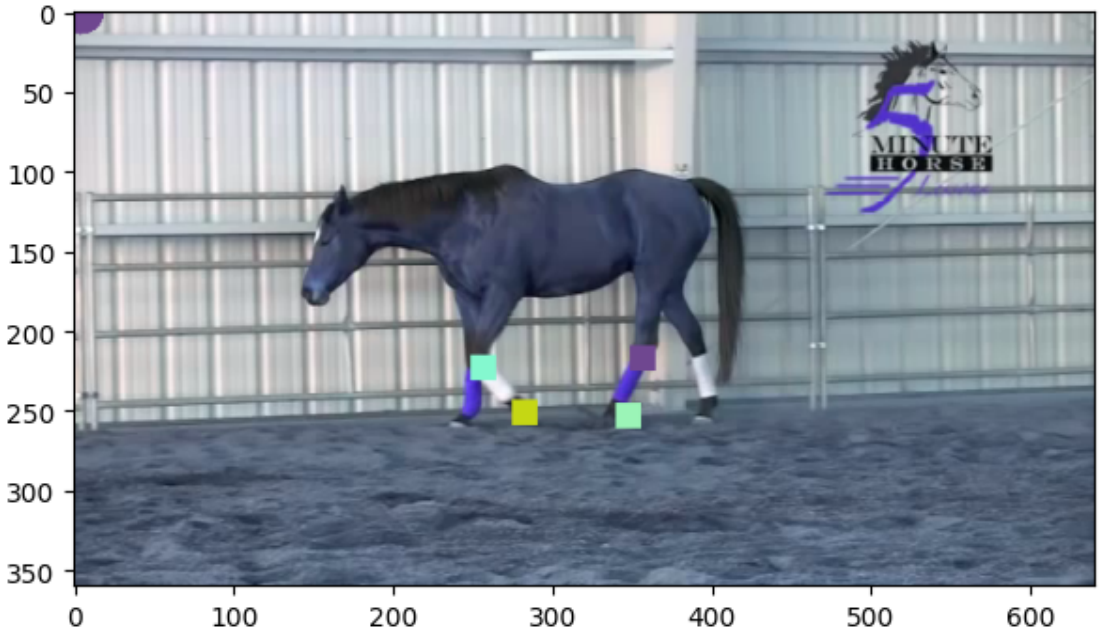
Epoch 510: Loss (2788.476806640625)

<Figure size 640x480 with 0 Axes>



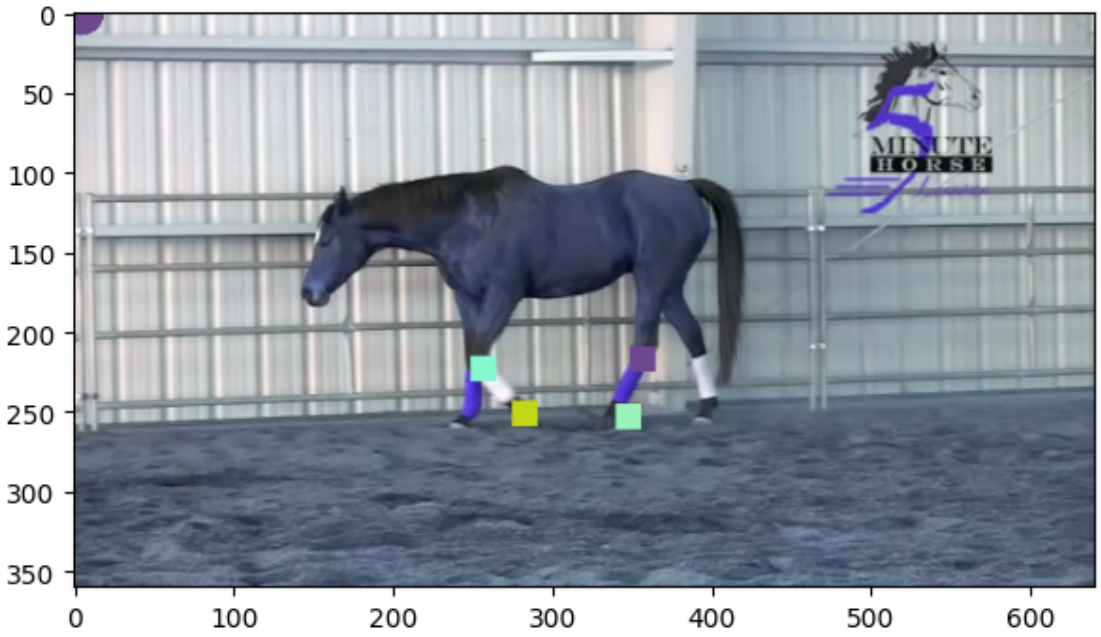
Epoch 520: Loss (2768.368408203125)

<Figure size 640x480 with 0 Axes>



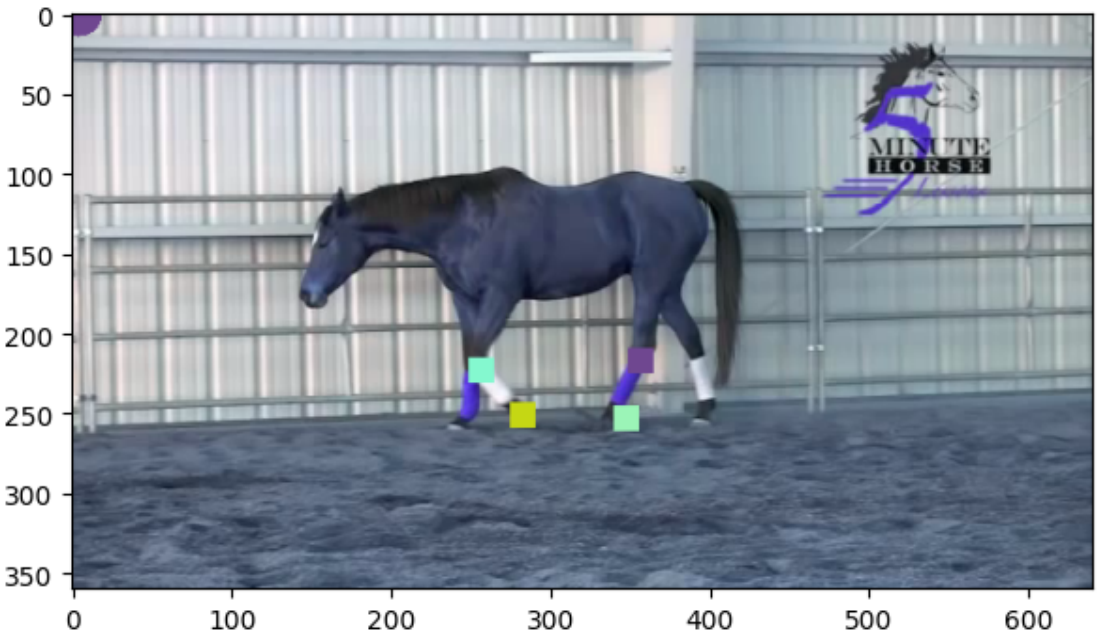
Epoch 530: Loss (2748.4248046875)

<Figure size 640x480 with 0 Axes>



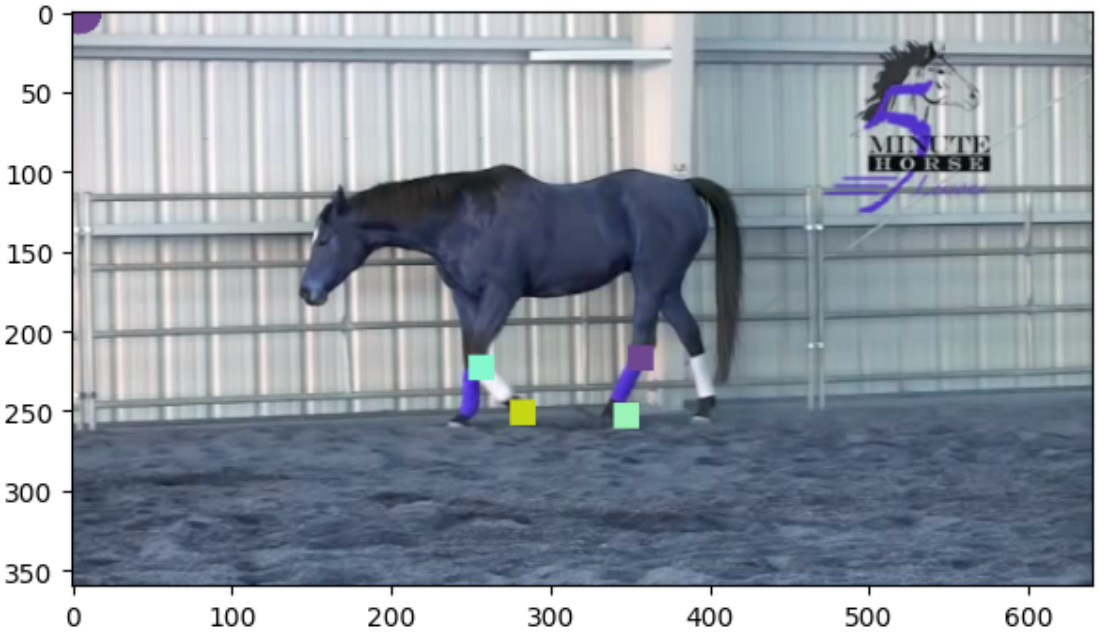
Epoch 540: Loss (2736.01416015625)

<Figure size 640x480 with 0 Axes>



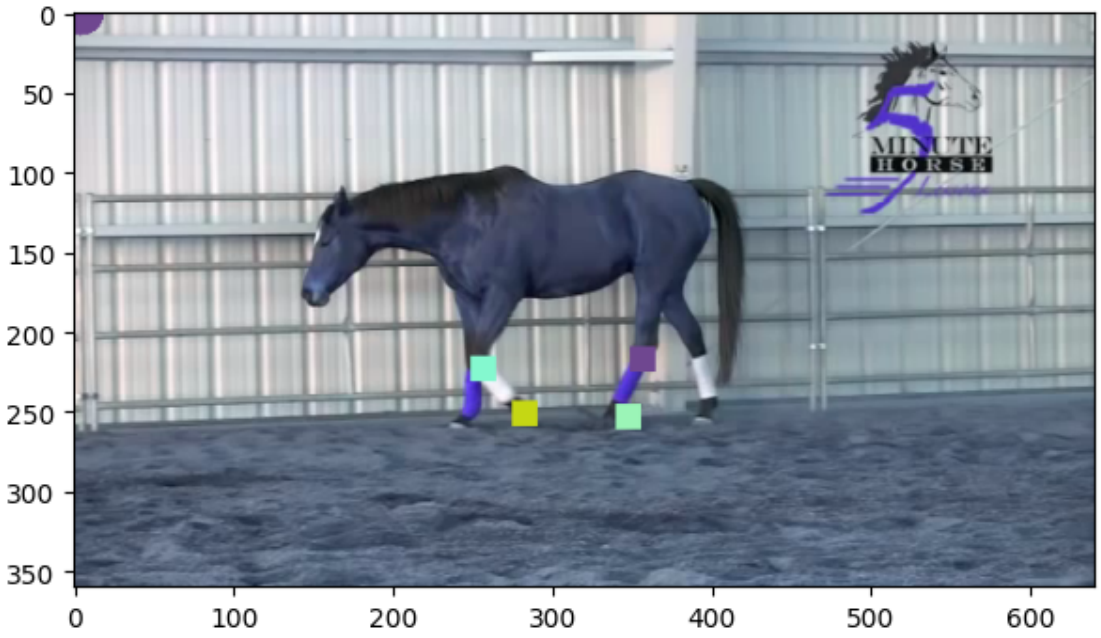
Epoch 550: Loss (2735.193115234375)

<Figure size 640x480 with 0 Axes>



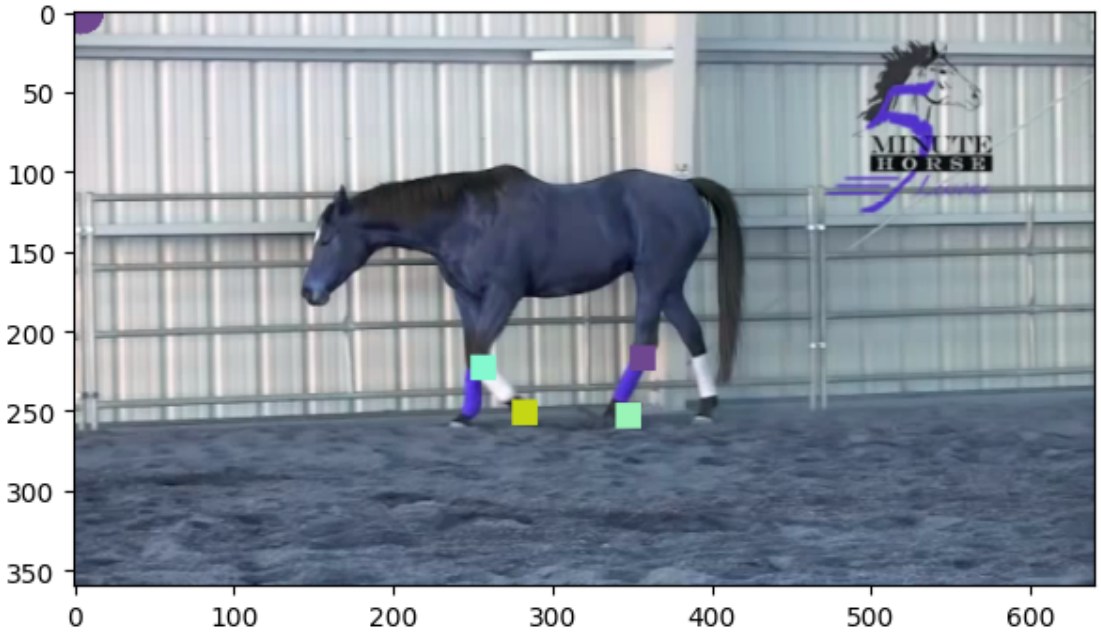
Epoch 560: Loss (2730.66650390625)

<Figure size 640x480 with 0 Axes>



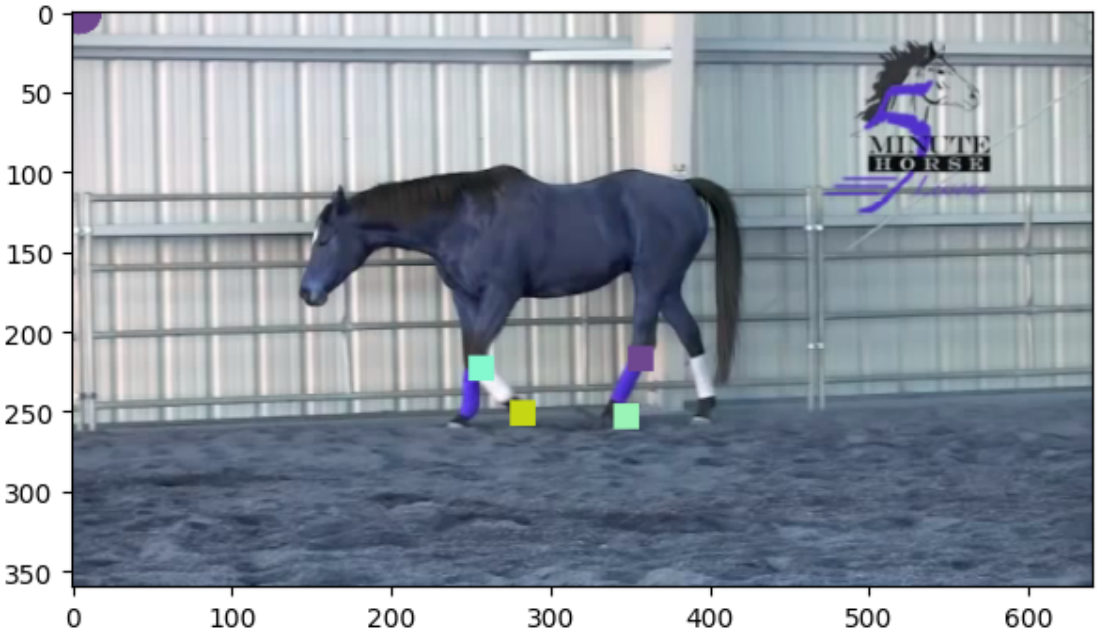
Epoch 570: Loss (2728.3388671875)

<Figure size 640x480 with 0 Axes>



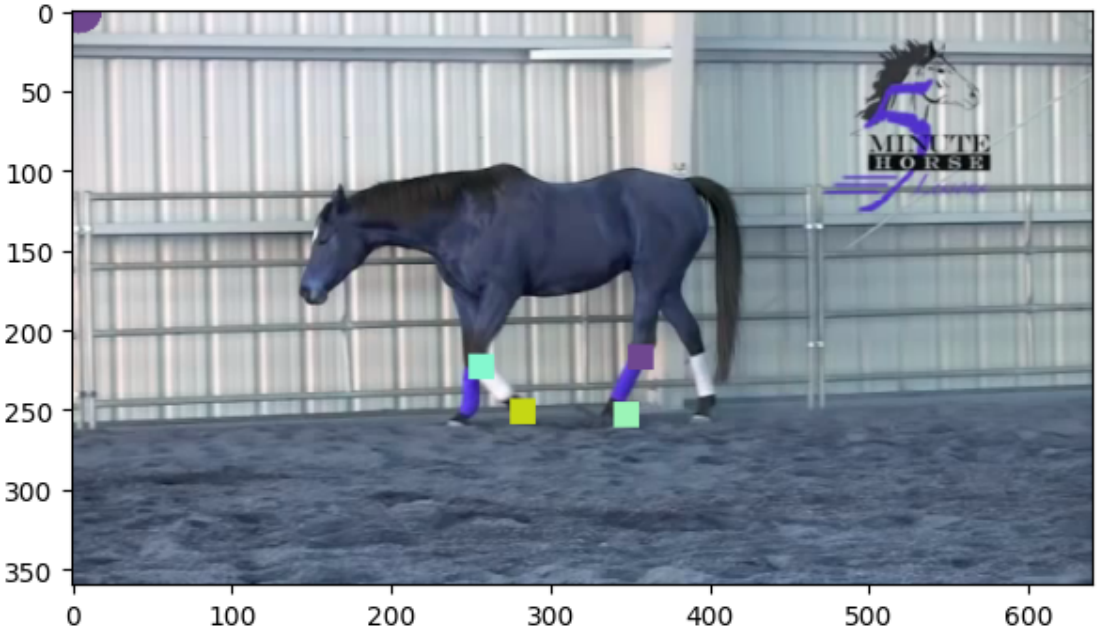
Epoch 580: Loss (2725.957275390625)

<Figure size 640x480 with 0 Axes>

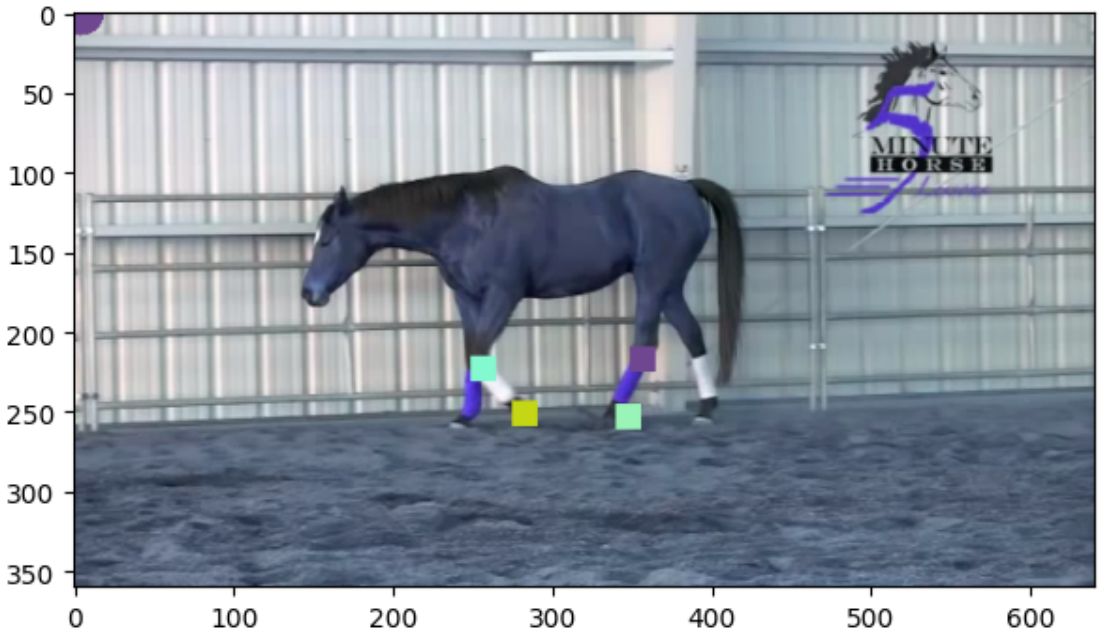


Epoch 590: Loss (2723.641845703125)

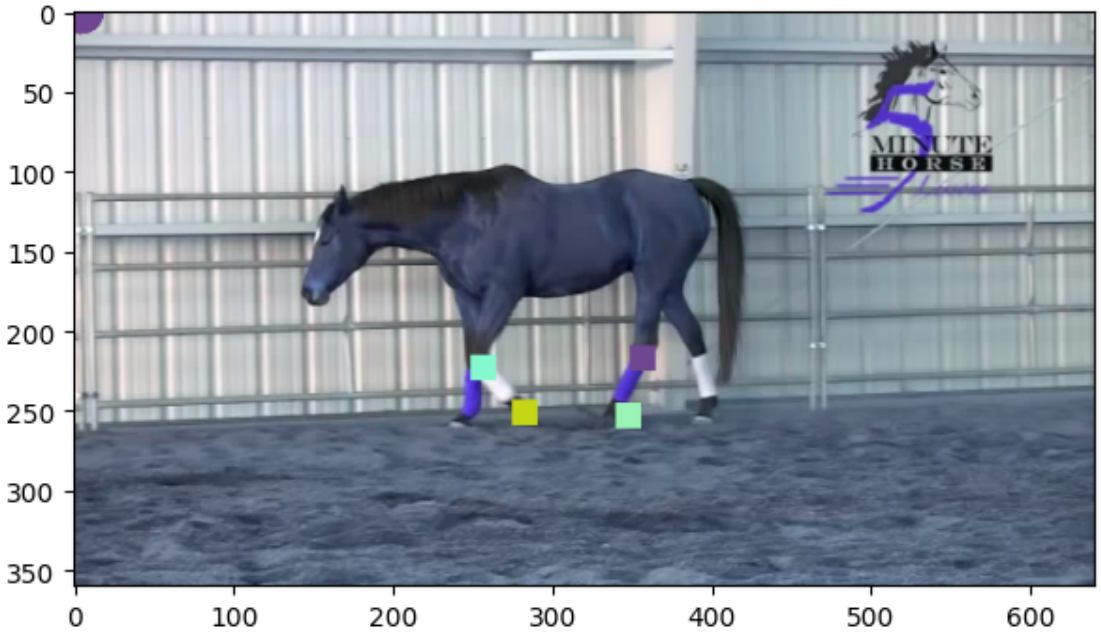
<Figure size 640x480 with 0 Axes>



```
Epoch 600: Loss (2721.40966796875)
<Figure size 640x480 with 0 Axes>
```

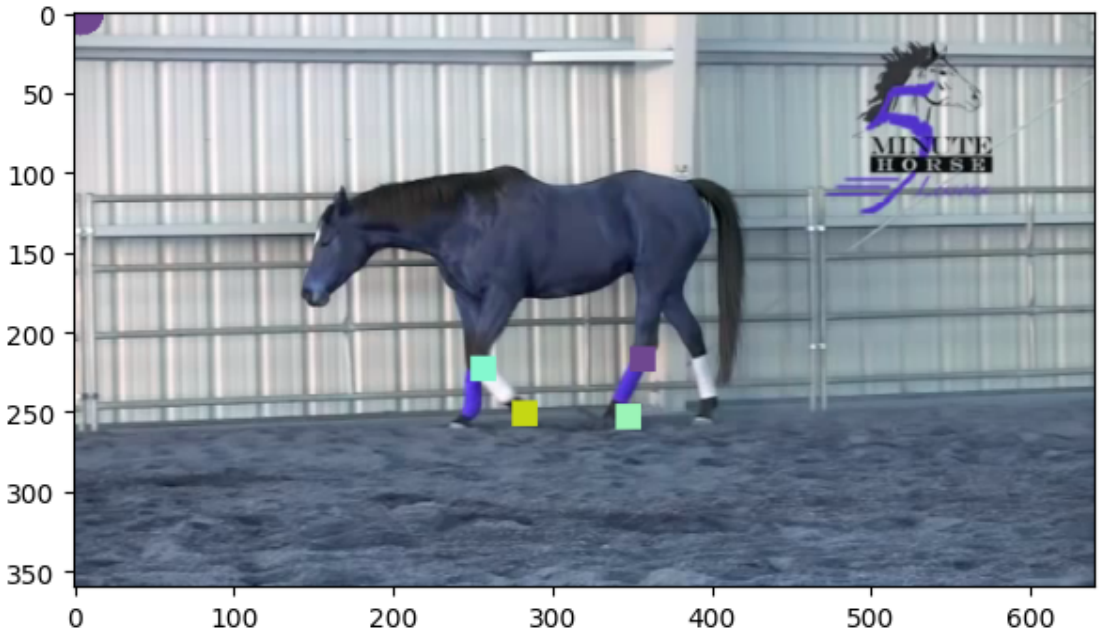



```
Epoch 610: Loss (2719.22119140625)
<Figure size 640x480 with 0 Axes>
```



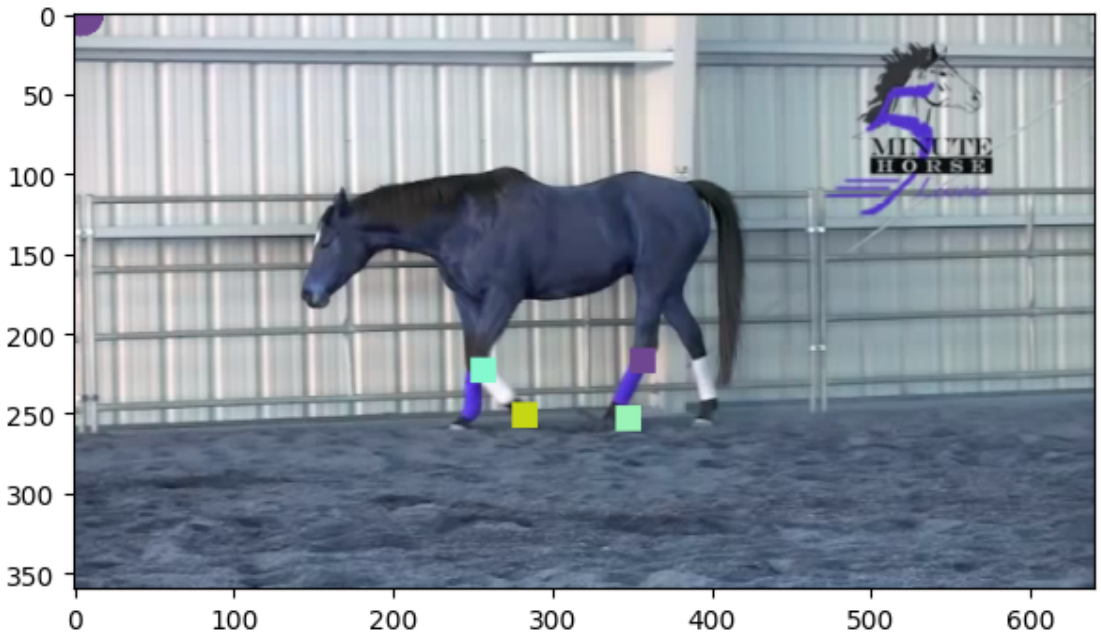
Epoch 620: Loss (2717.07470703125)

<Figure size 640x480 with 0 Axes>



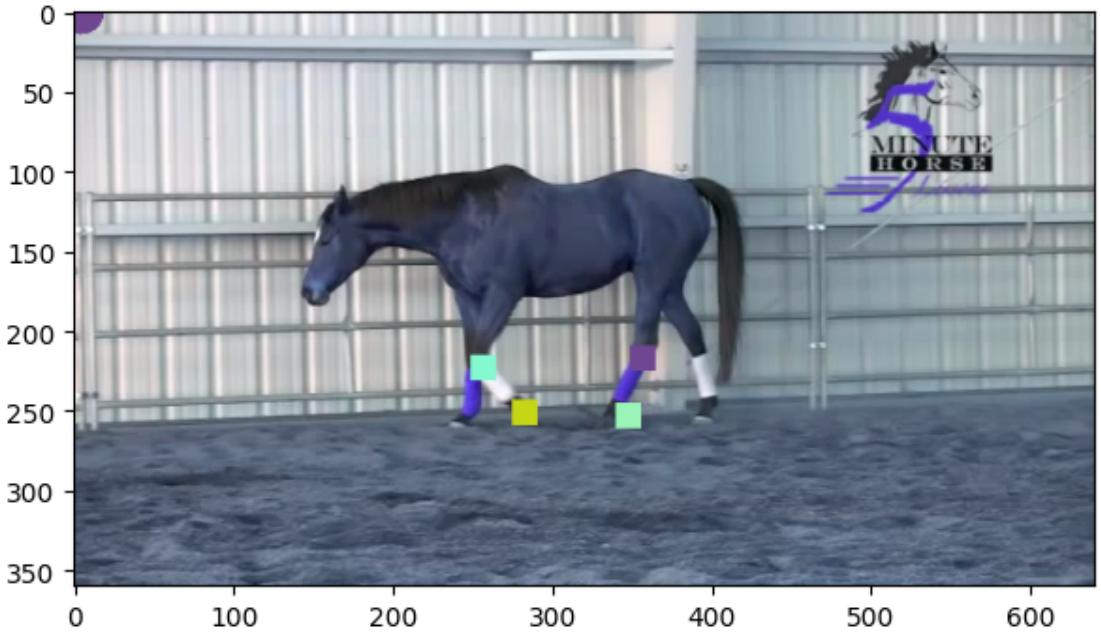
Epoch 630: Loss (2714.9638671875)

<Figure size 640x480 with 0 Axes>



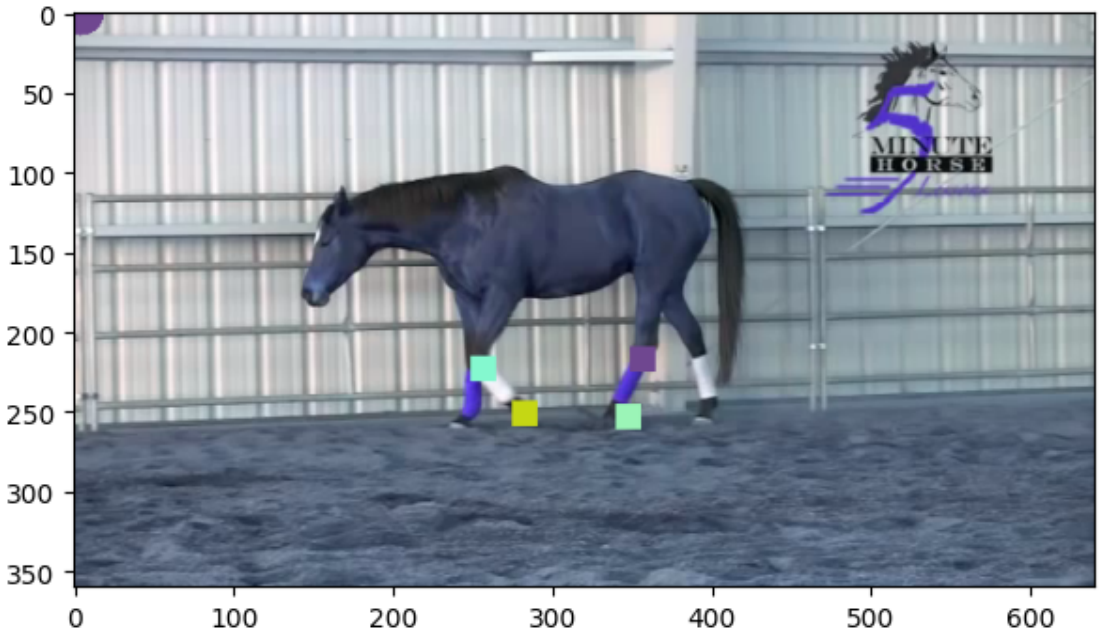
Epoch 640: Loss (2712.885009765625)

<Figure size 640x480 with 0 Axes>



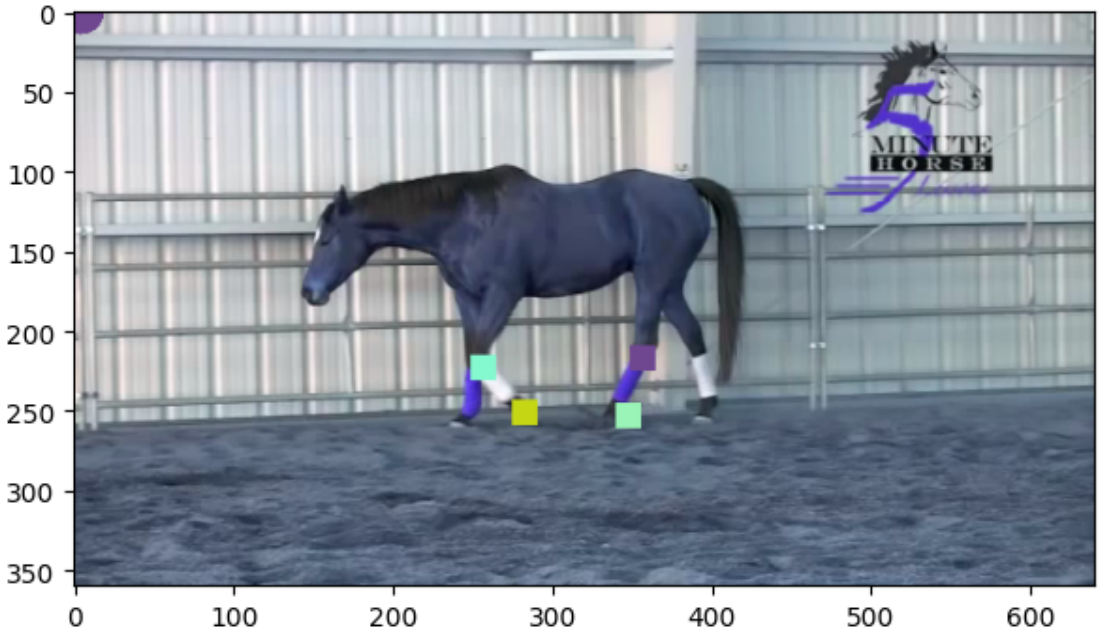
Epoch 650: Loss (2710.8359375)

<Figure size 640x480 with 0 Axes>



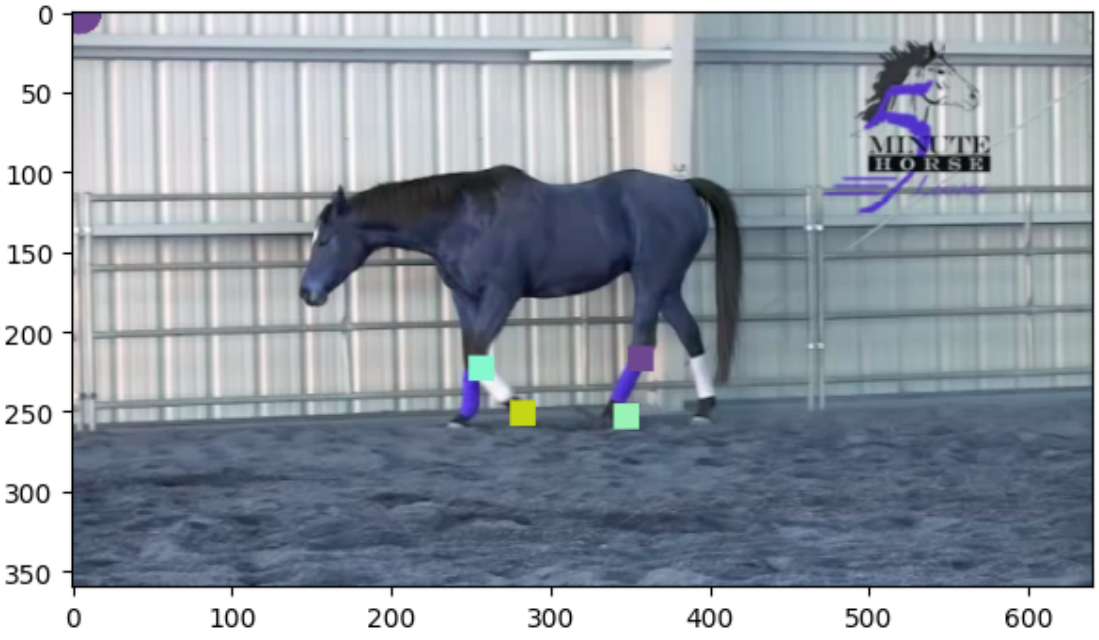
Epoch 660: Loss (2708.814208984375)

<Figure size 640x480 with 0 Axes>



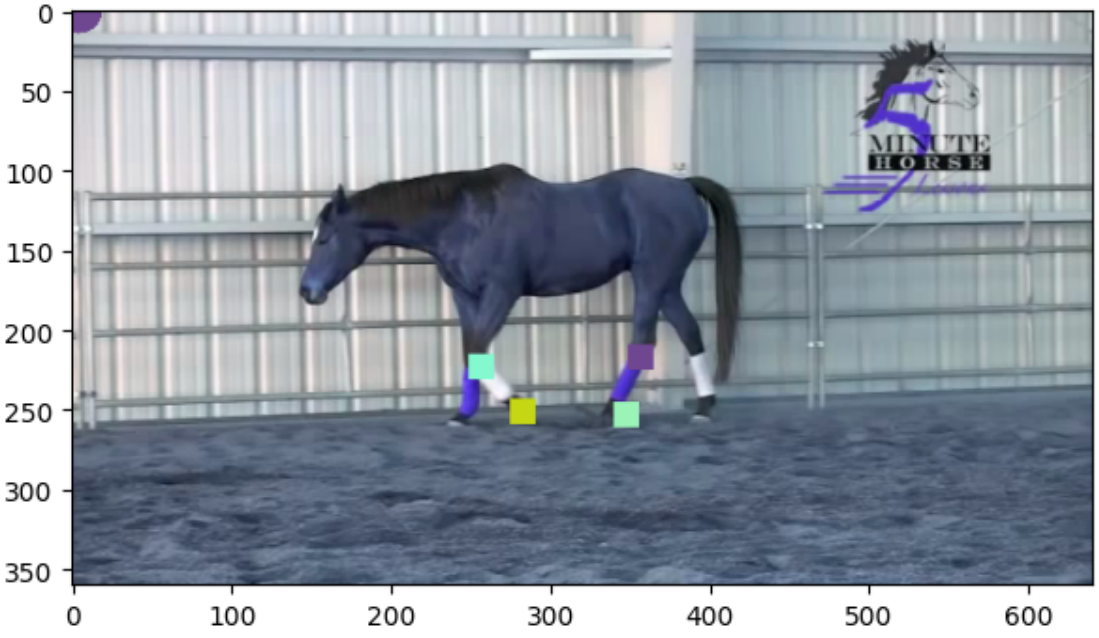
Epoch 670: Loss (2706.81787109375)

<Figure size 640x480 with 0 Axes>

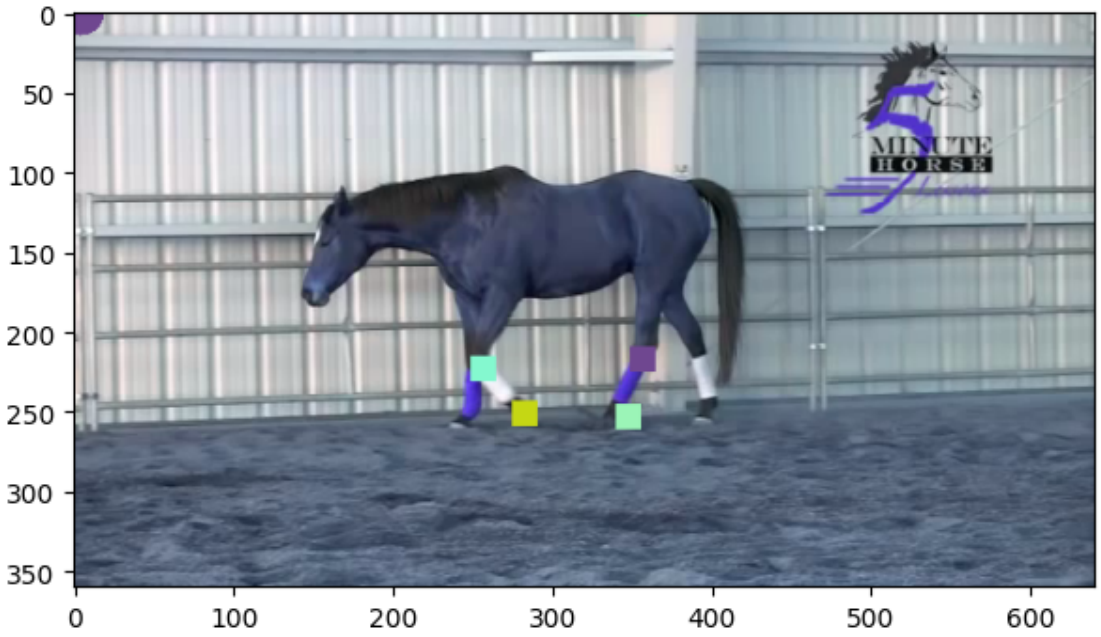


Epoch 680: Loss (2704.845458984375)

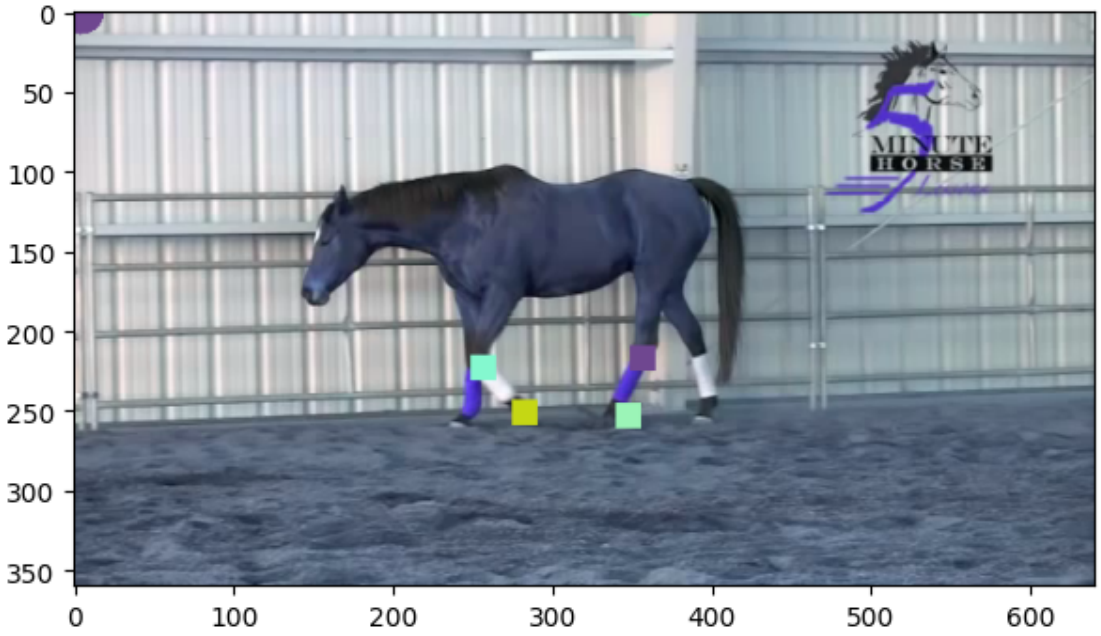
<Figure size 640x480 with 0 Axes>



```
Epoch 690: Loss (2702.89453125)
<Figure size 640x480 with 0 Axes>
```

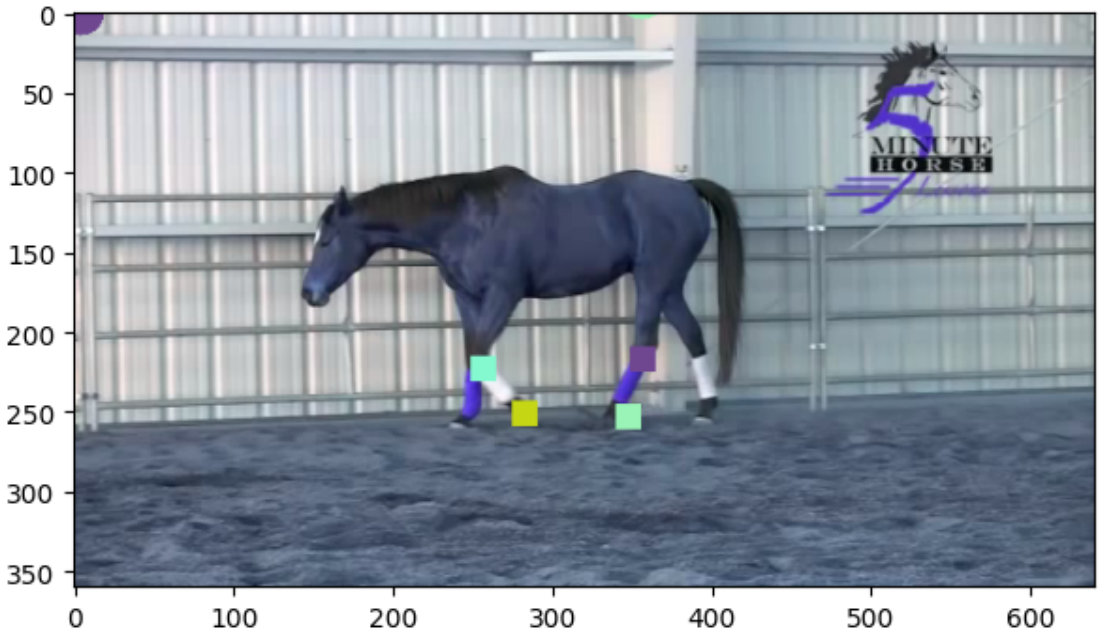



```
Epoch 700: Loss (2700.964599609375)
<Figure size 640x480 with 0 Axes>
```



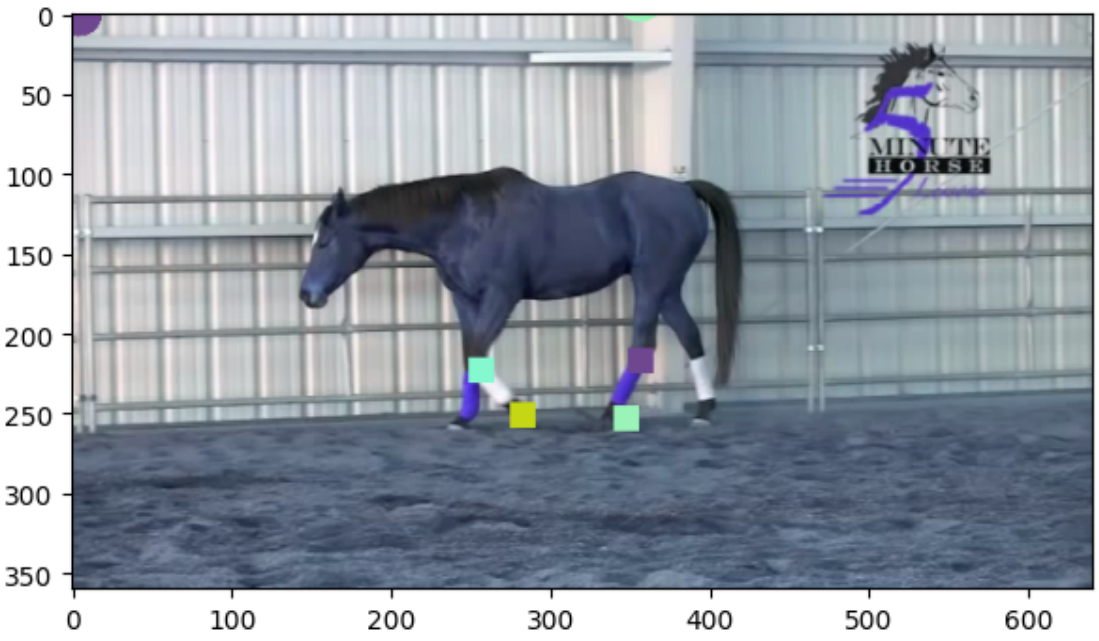
Epoch 710: Loss (2699.053955078125)

<Figure size 640x480 with 0 Axes>



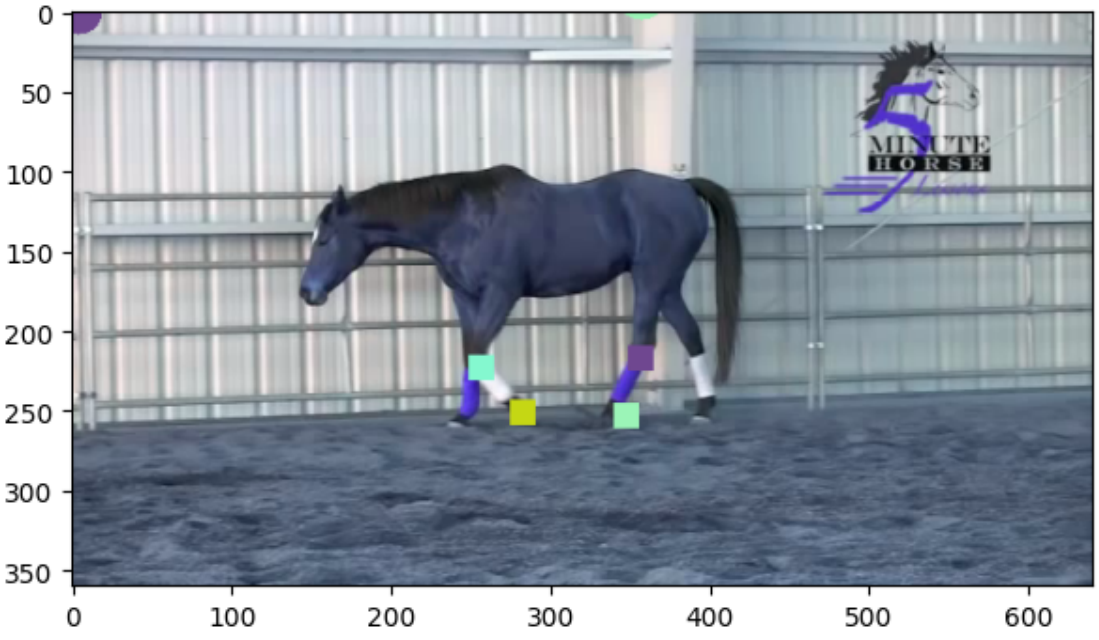
Epoch 720: Loss (2697.160888671875)

<Figure size 640x480 with 0 Axes>



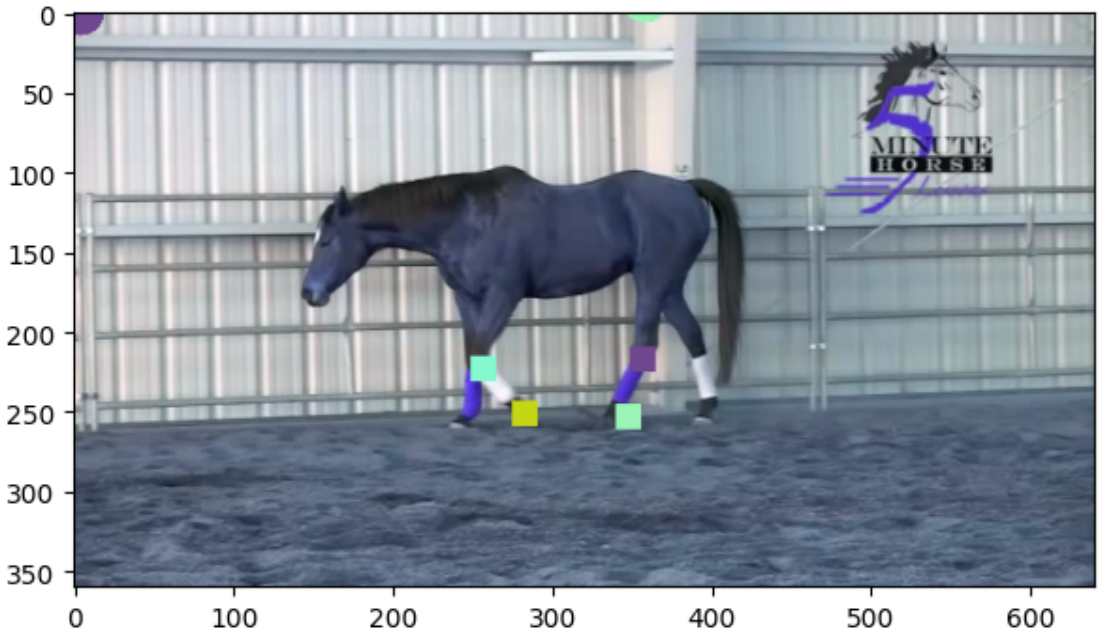
Epoch 730: Loss (2695.28515625)

<Figure size 640x480 with 0 Axes>



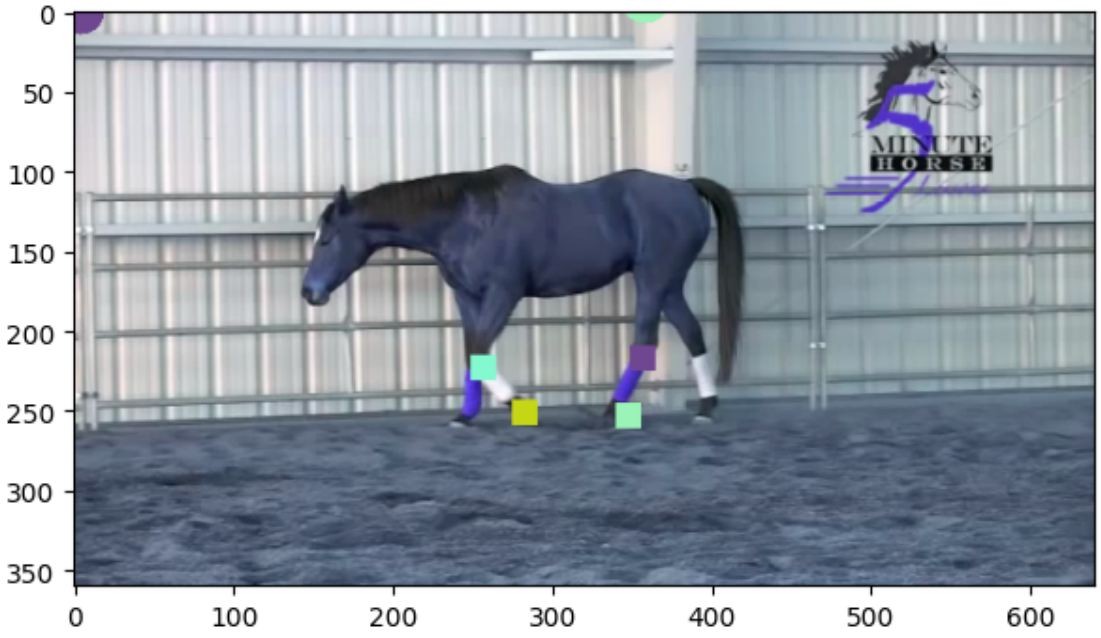
Epoch 740: Loss (2693.425048828125)

<Figure size 640x480 with 0 Axes>



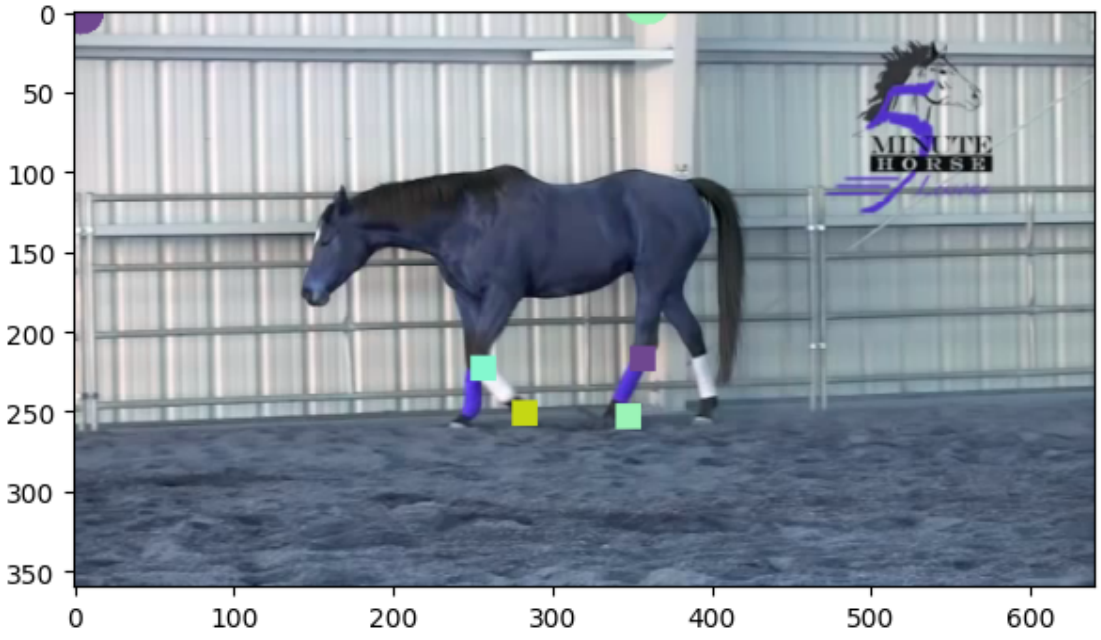
Epoch 750: Loss (2691.579833984375)

<Figure size 640x480 with 0 Axes>



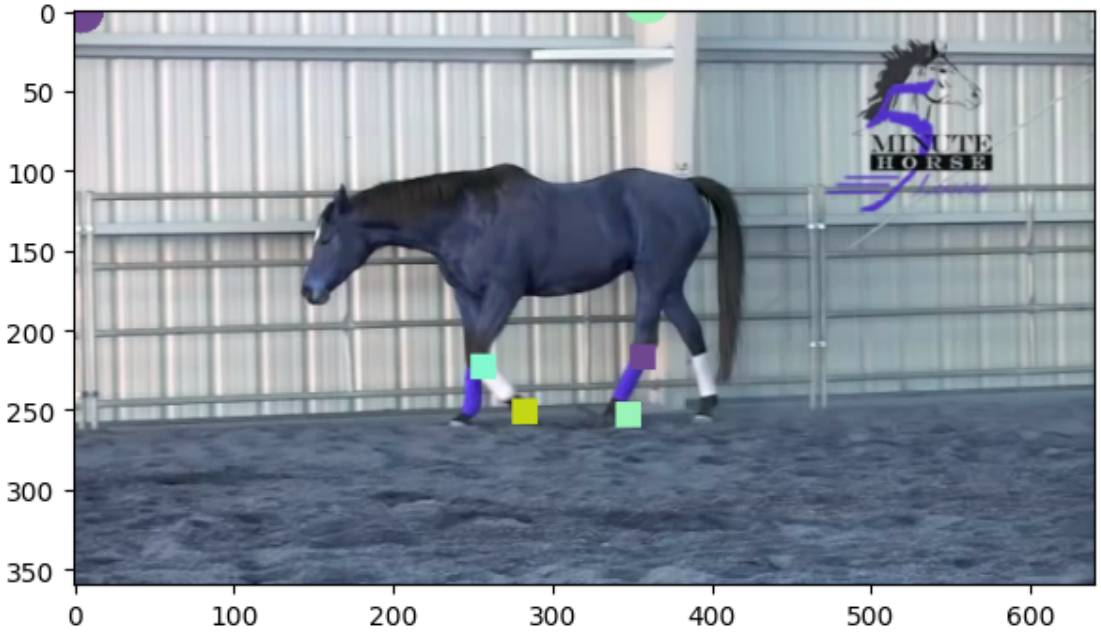
Epoch 760: Loss (2689.7490234375)

<Figure size 640x480 with 0 Axes>



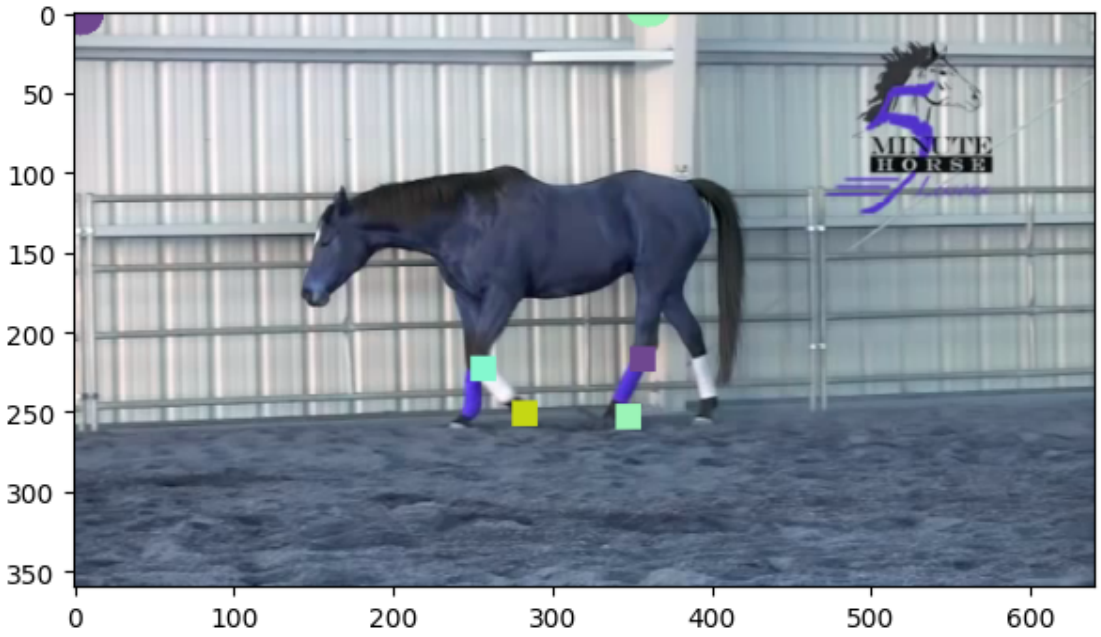
Epoch 770: Loss (2687.930908203125)

<Figure size 640x480 with 0 Axes>



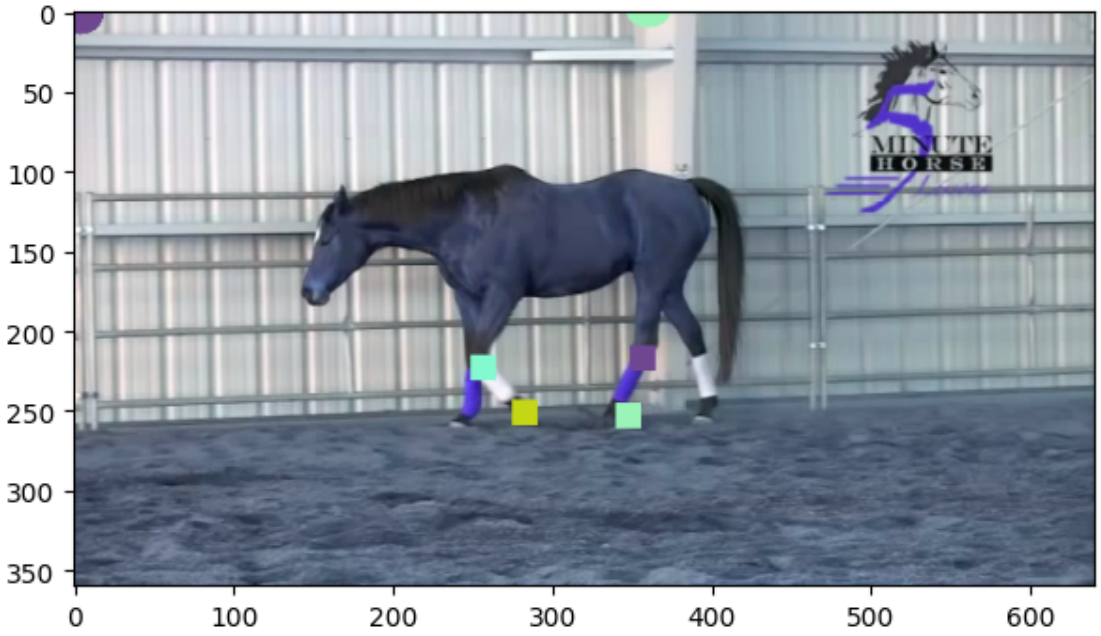
Epoch 780: Loss (2686.125244140625)

<Figure size 640x480 with 0 Axes>



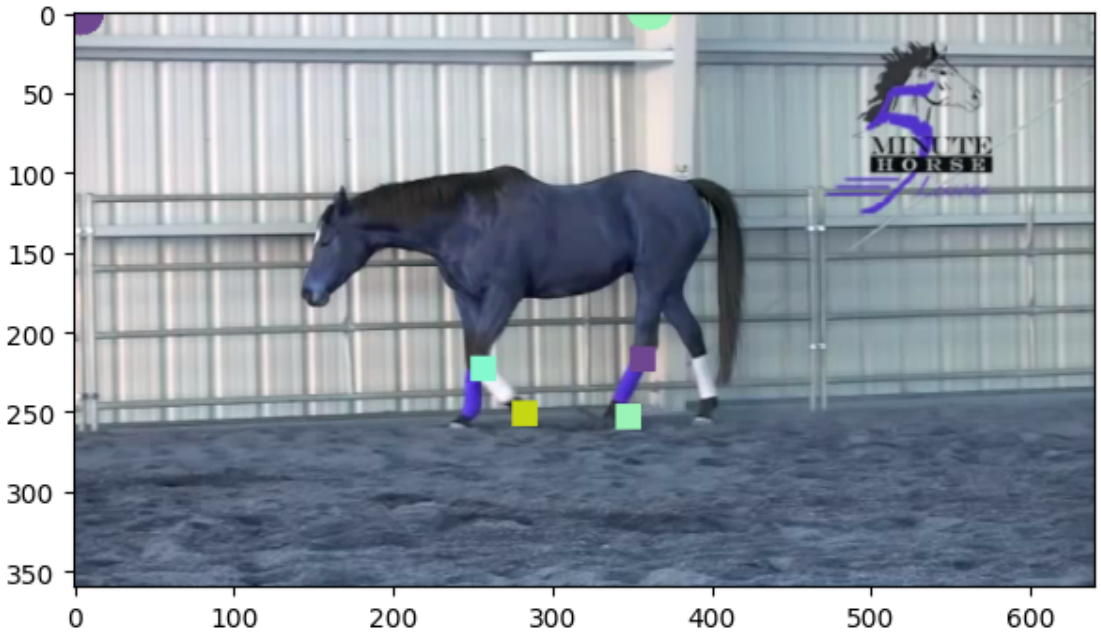
Epoch 790: Loss (2684.331298828125)

<Figure size 640x480 with 0 Axes>



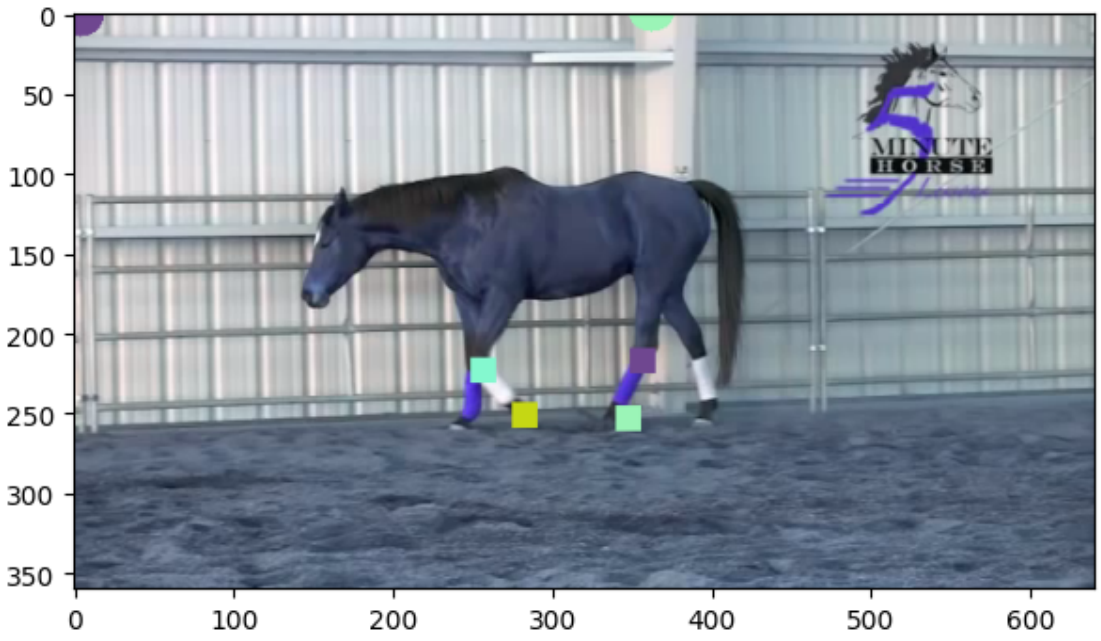
Epoch 800: Loss (2682.54833984375)

<Figure size 640x480 with 0 Axes>



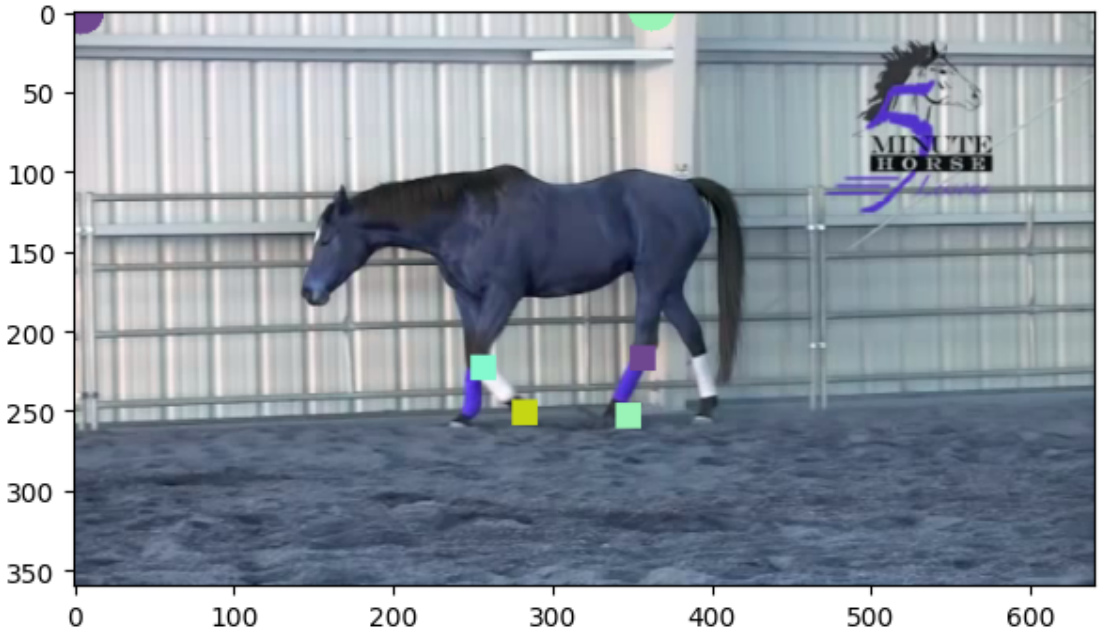
Epoch 810: Loss (2680.775634765625)

<Figure size 640x480 with 0 Axes>



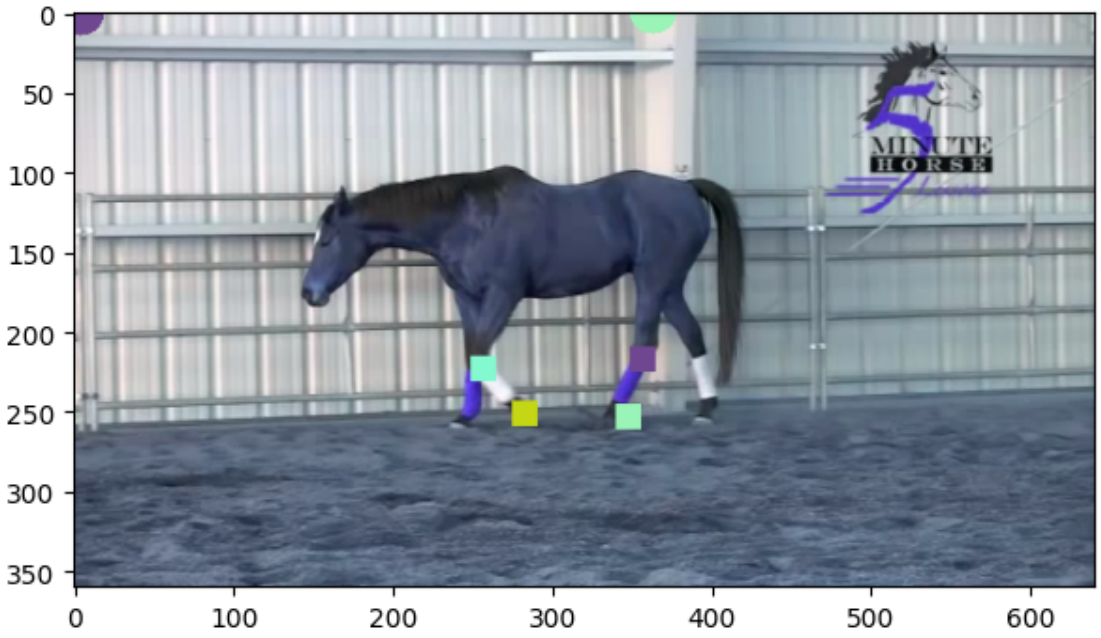
Epoch 820: Loss (2679.012939453125)

<Figure size 640x480 with 0 Axes>



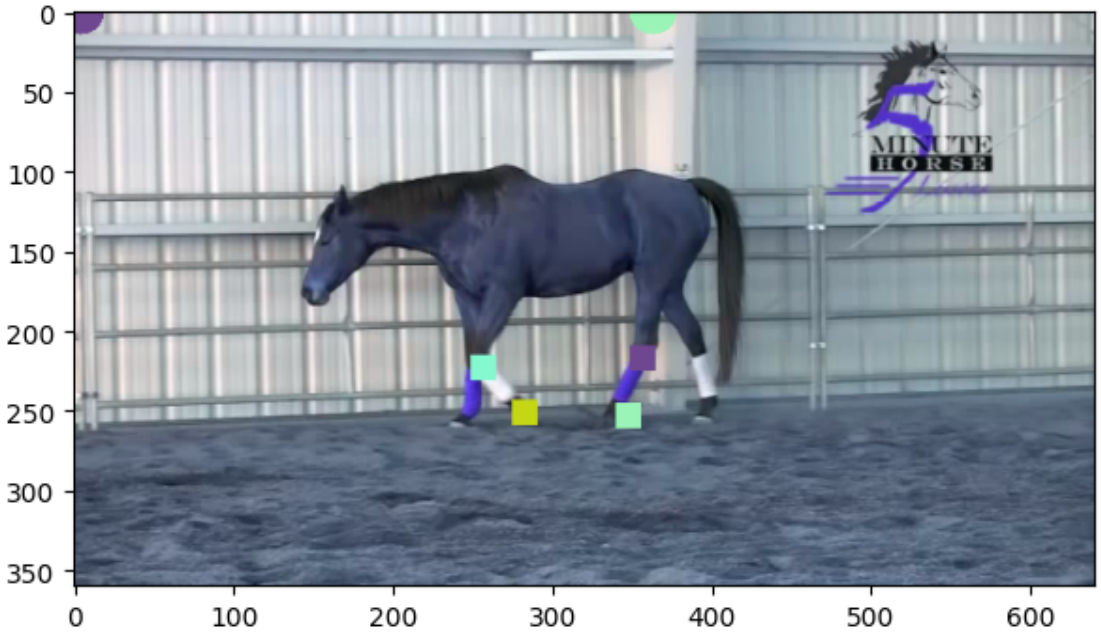
Epoch 830: Loss (2677.259033203125)

<Figure size 640x480 with 0 Axes>



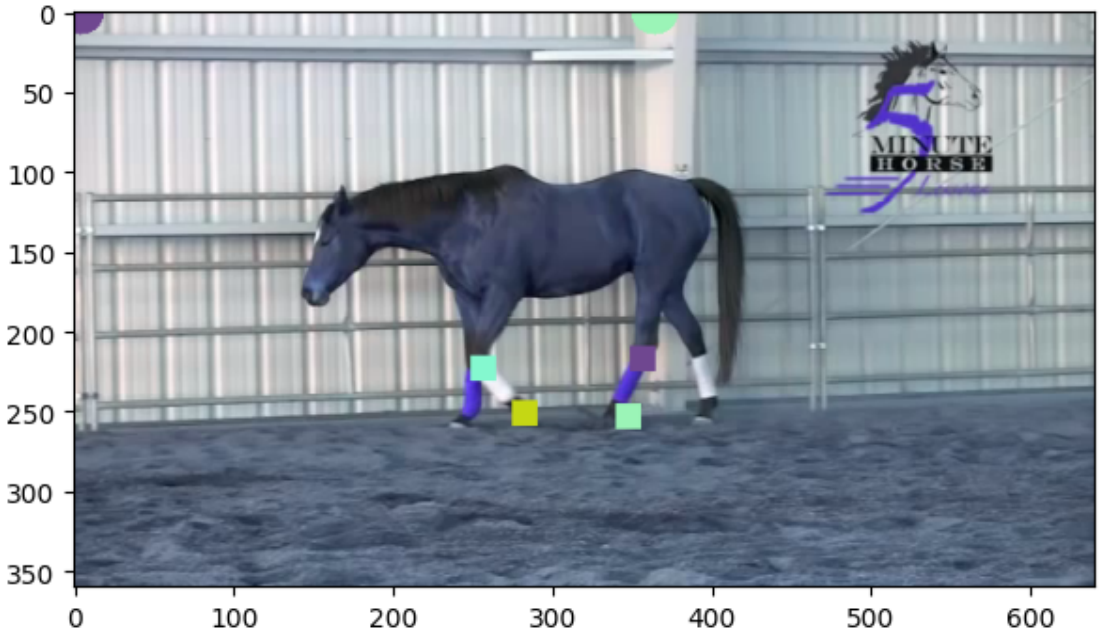
Epoch 840: Loss (2675.513916015625)

<Figure size 640x480 with 0 Axes>



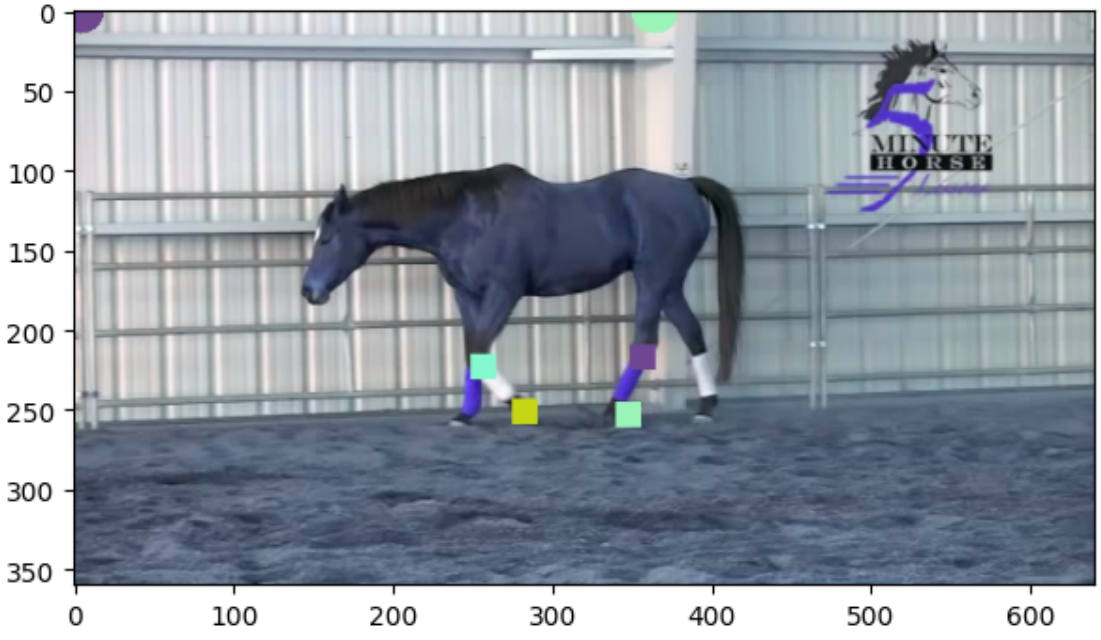
Epoch 850: Loss (2673.77685546875)

<Figure size 640x480 with 0 Axes>



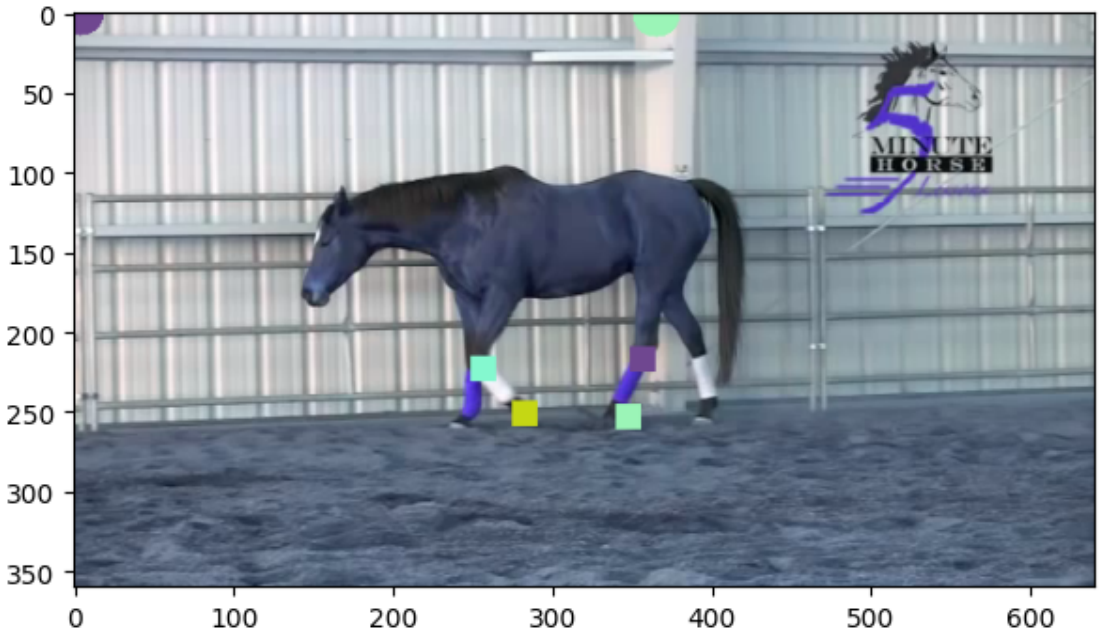
Epoch 860: Loss (2672.04736328125)

<Figure size 640x480 with 0 Axes>



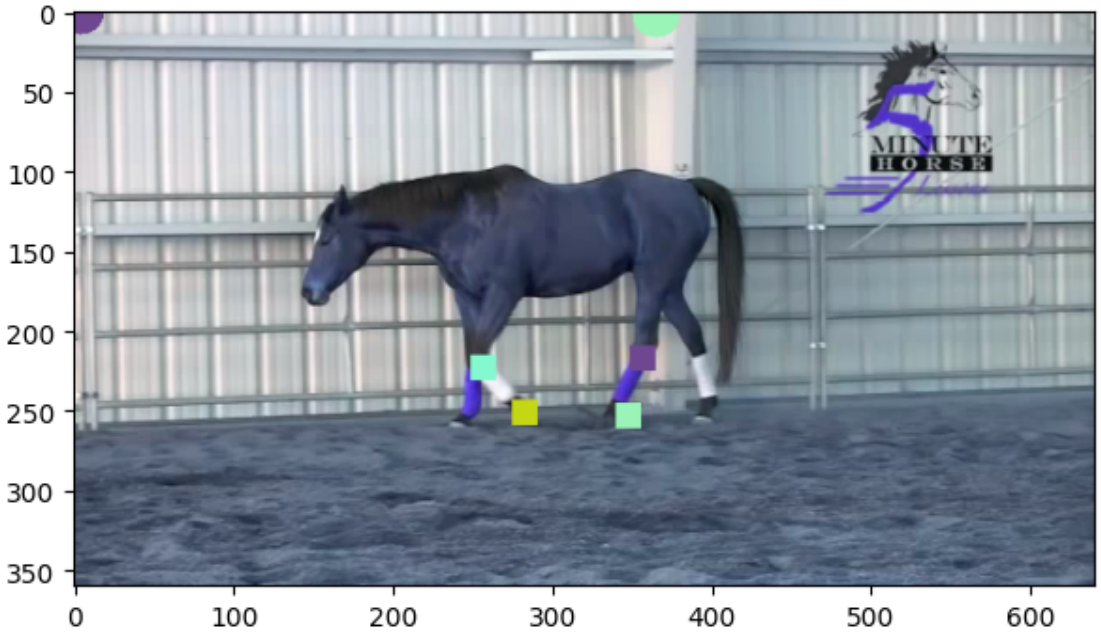
Epoch 870: Loss (2670.3251953125)

<Figure size 640x480 with 0 Axes>



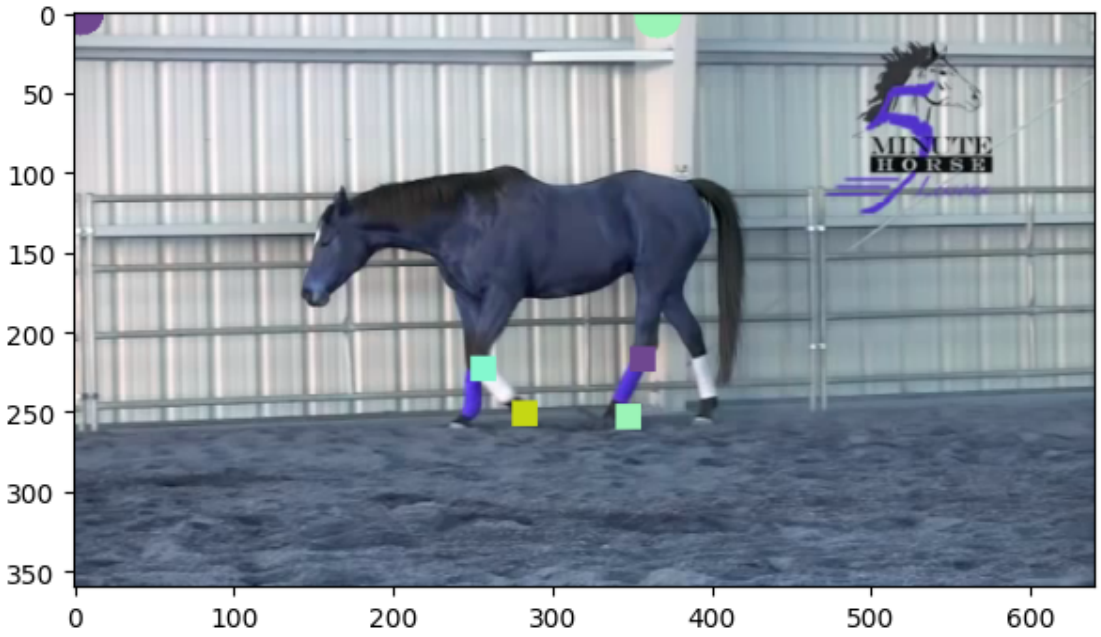
Epoch 880: Loss (2668.609375)

<Figure size 640x480 with 0 Axes>



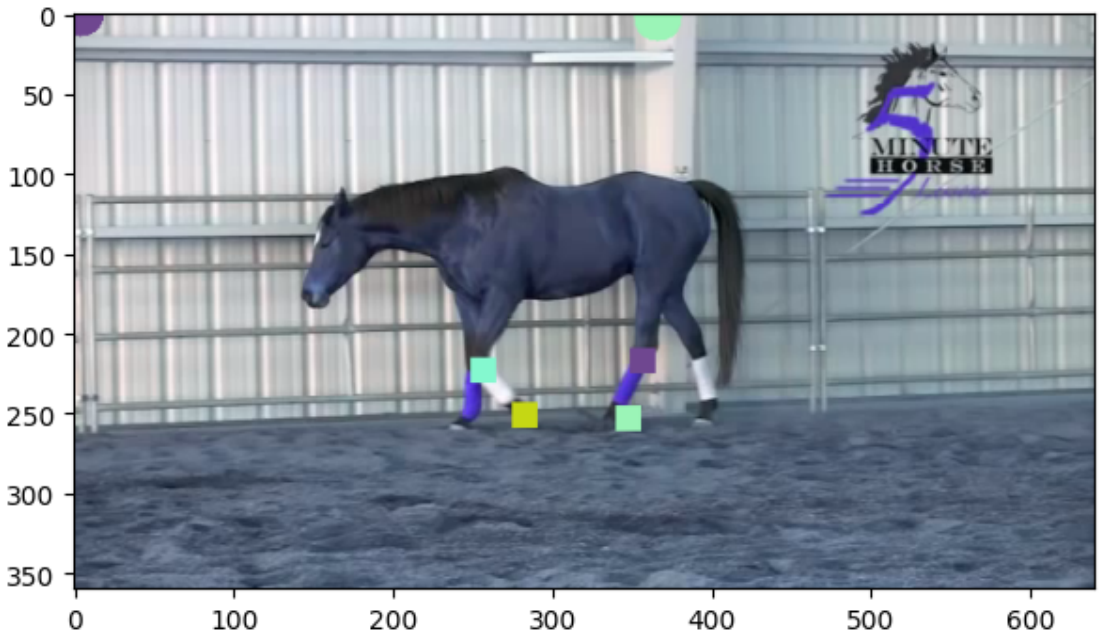
Epoch 890: Loss (2666.900146484375)

<Figure size 640x480 with 0 Axes>

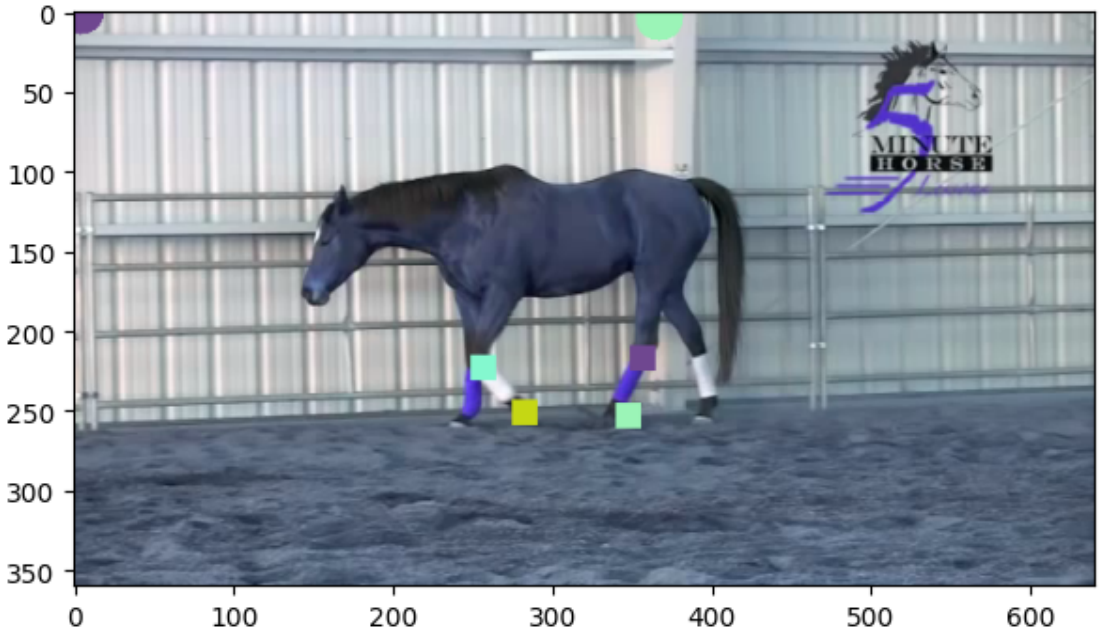


Epoch 900: Loss (2665.196533203125)

<Figure size 640x480 with 0 Axes>

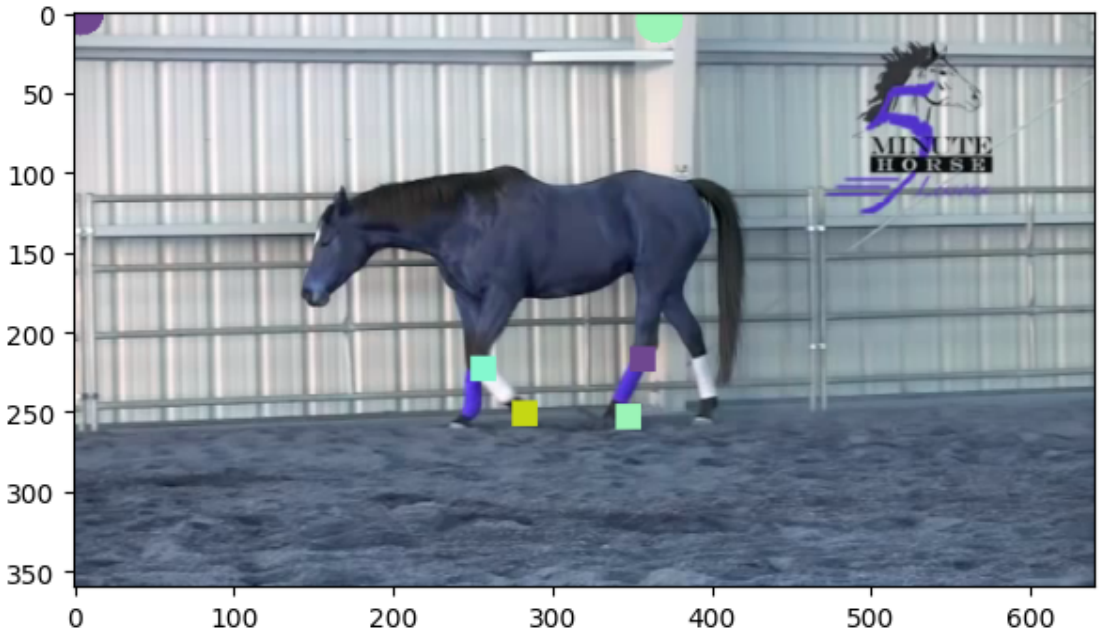



```
Epoch 910: Loss (2663.498291015625)
<Figure size 640x480 with 0 Axes>
```



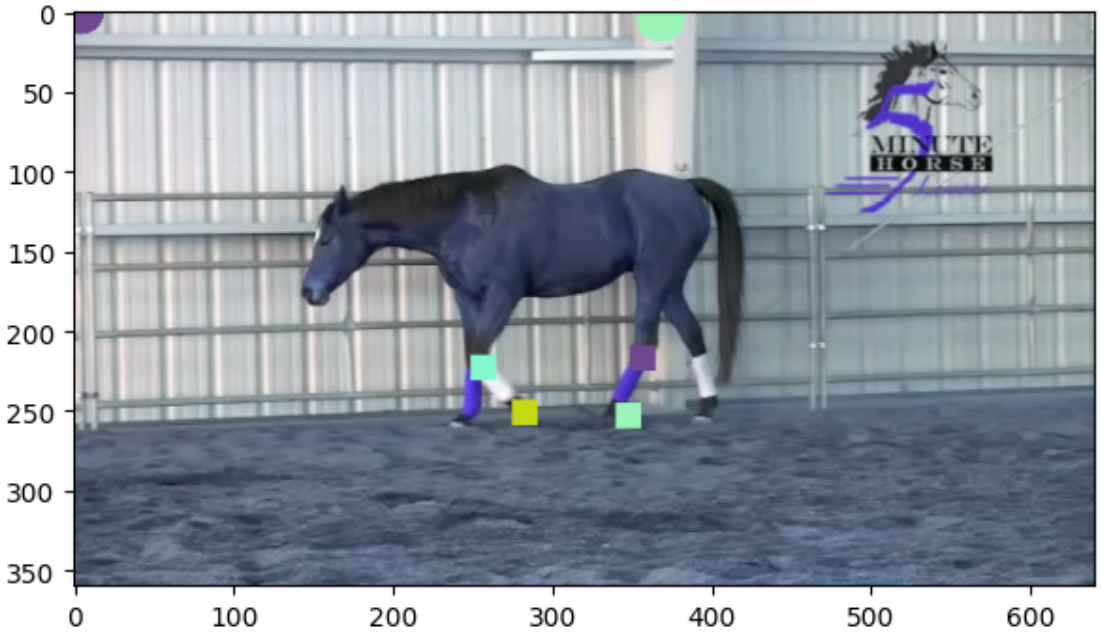

```
Epoch 920: Loss (2661.80517578125)
```

<Figure size 640x480 with 0 Axes>



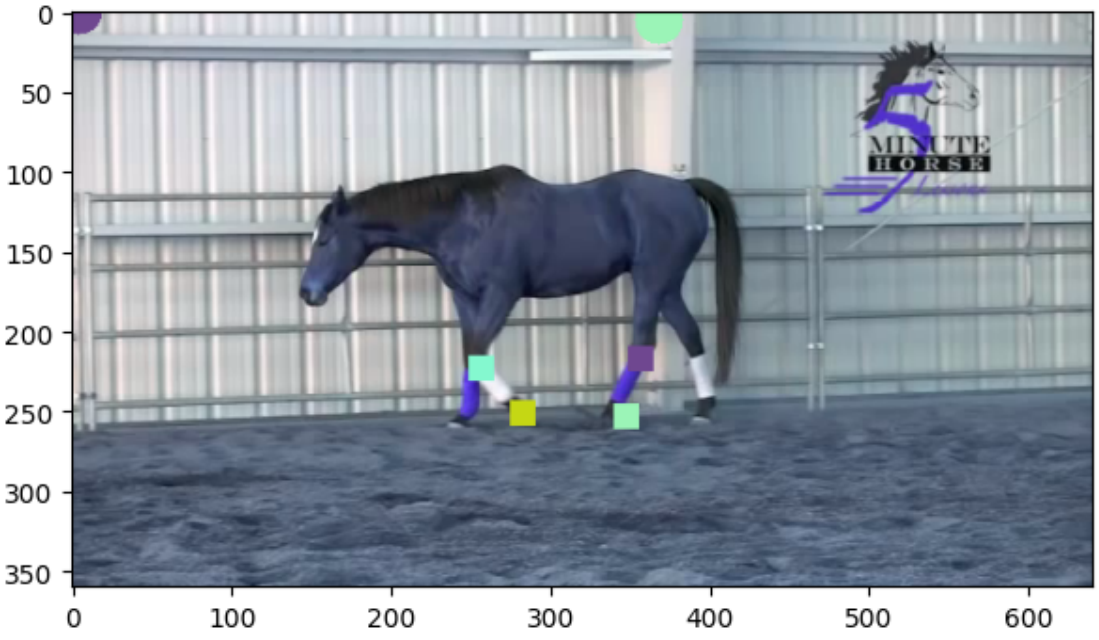
Epoch 930: Loss (2660.116943359375)

<Figure size 640x480 with 0 Axes>



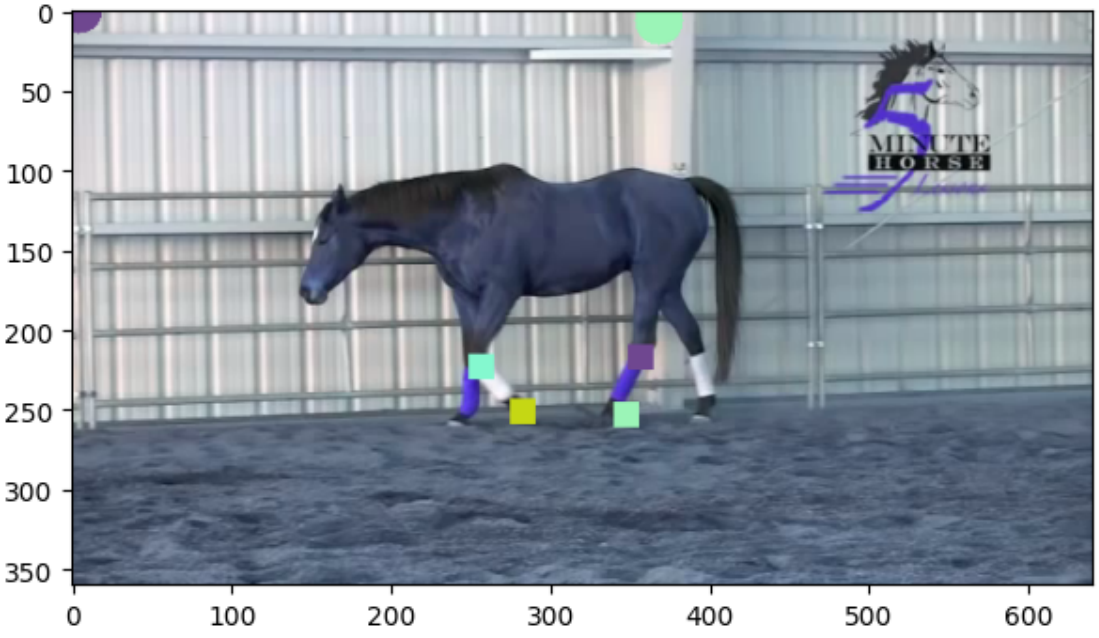
Epoch 940: Loss (2658.4326171875)

<Figure size 640x480 with 0 Axes>

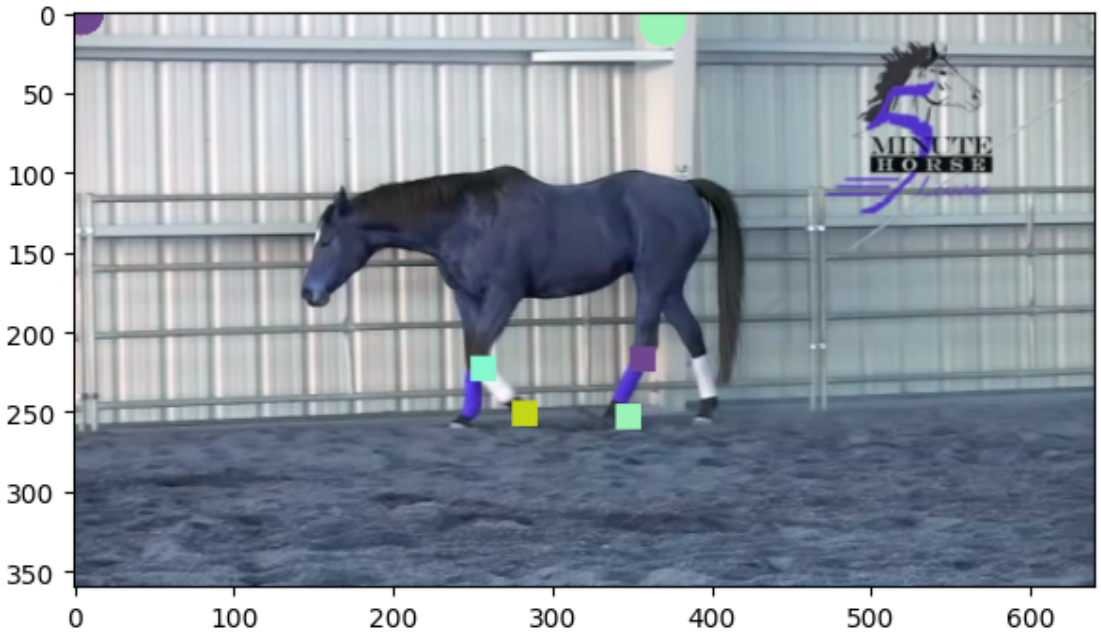


Epoch 950: Loss (2656.75244140625)

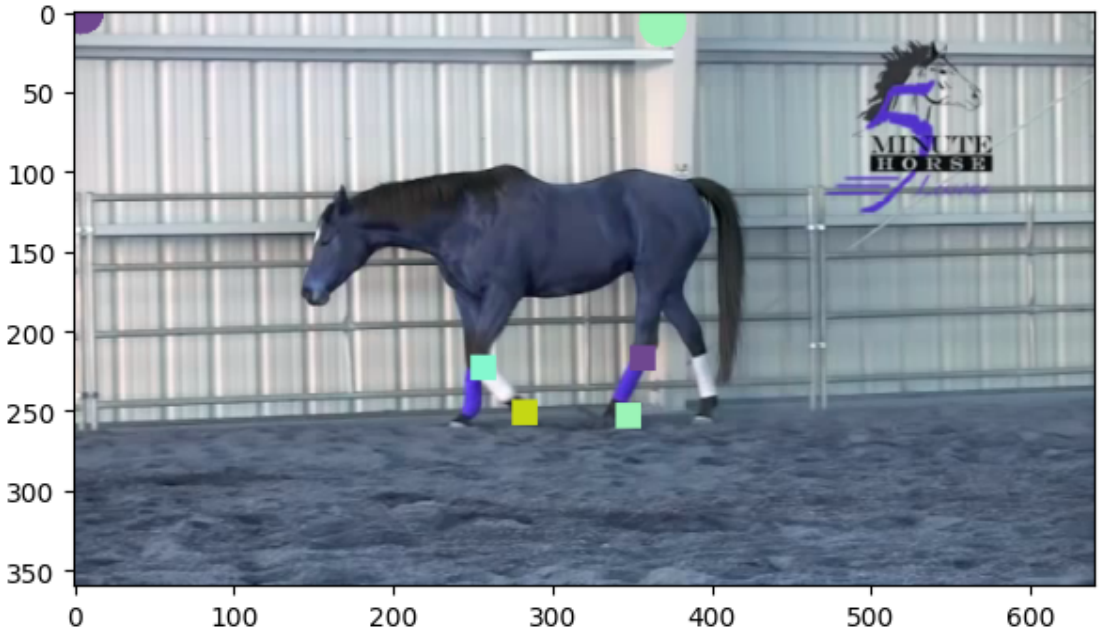
<Figure size 640x480 with 0 Axes>



```
Epoch 960: Loss (2655.075927734375)
<Figure size 640x480 with 0 Axes>
```

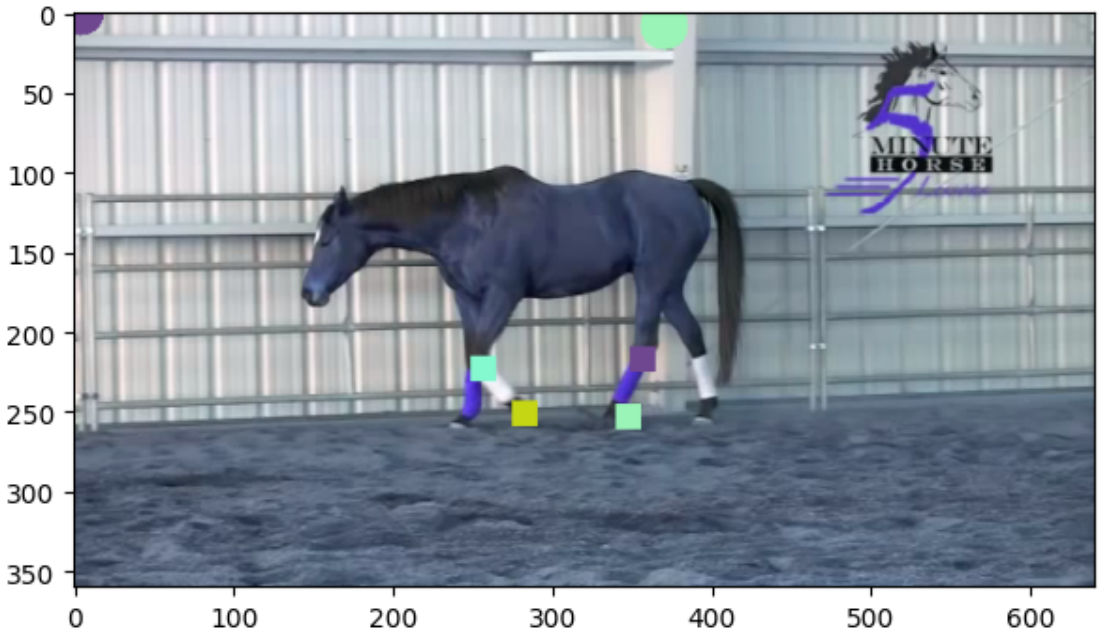



```
Epoch 970: Loss (2653.40283203125)
<Figure size 640x480 with 0 Axes>
```



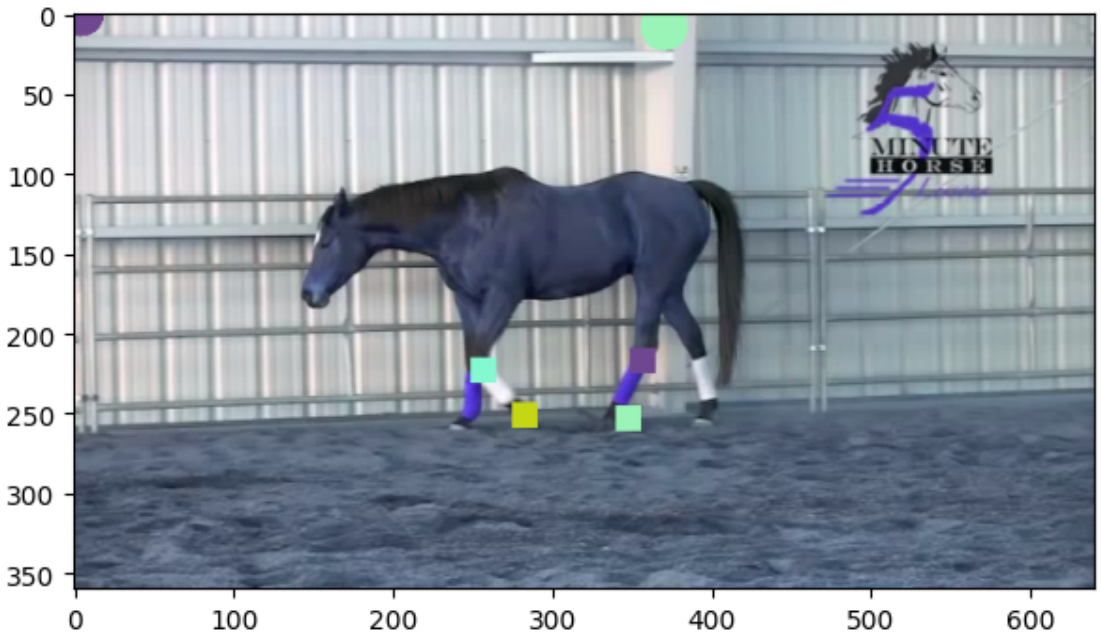
Epoch 980: Loss (2651.732177734375)

<Figure size 640x480 with 0 Axes>

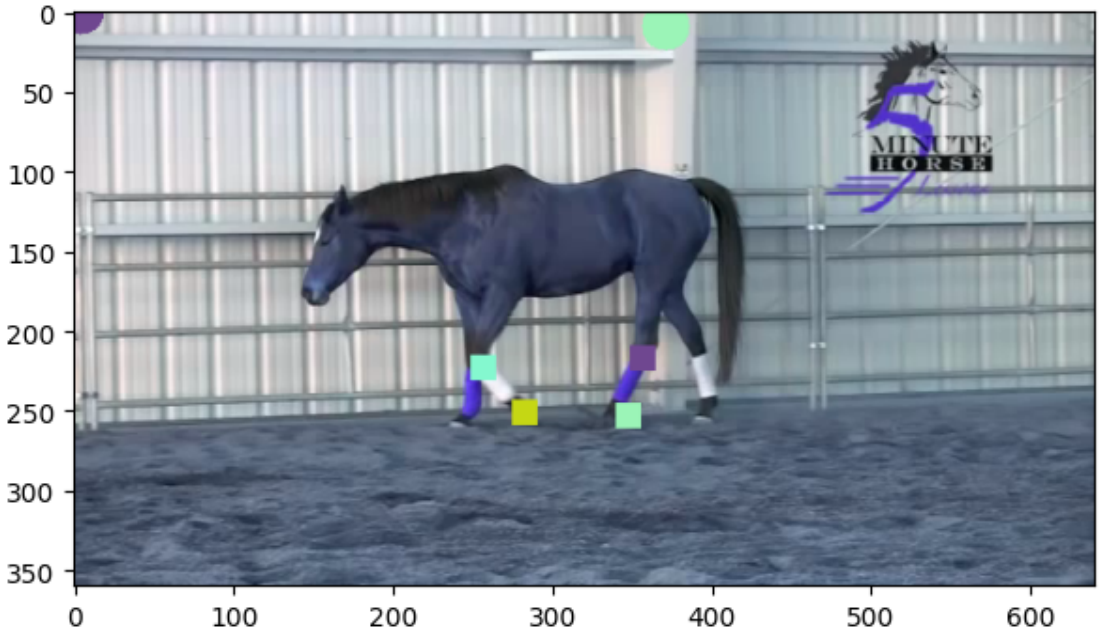


Epoch 990: Loss (2650.064697265625)

<Figure size 640x480 with 0 Axes>

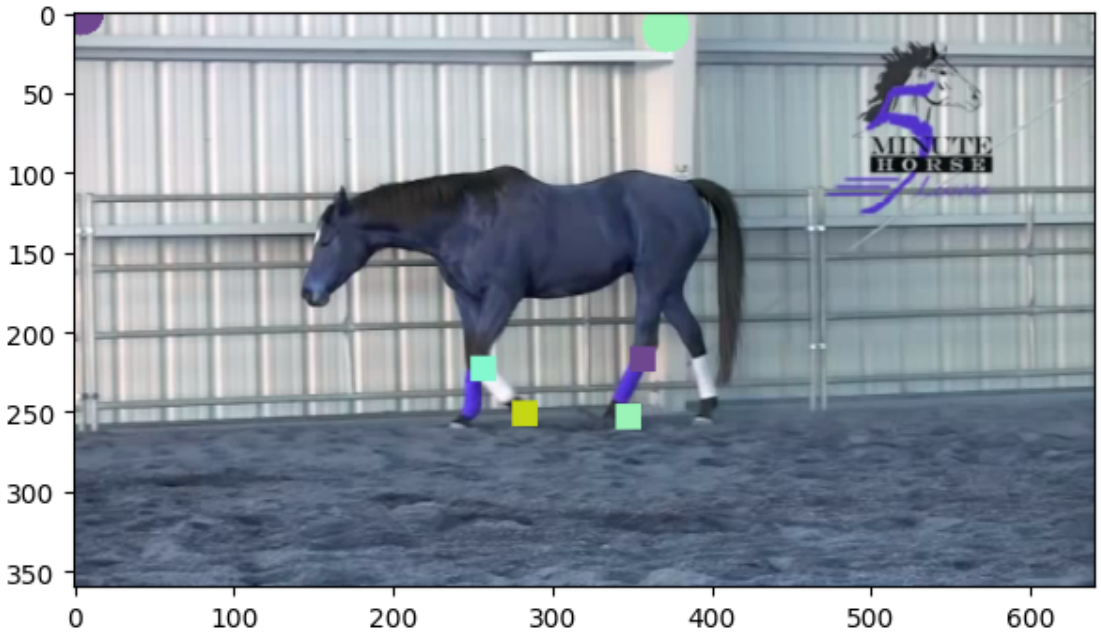



```
Epoch 1000: Loss (2648.3994140625)
<Figure size 640x480 with 0 Axes>
```



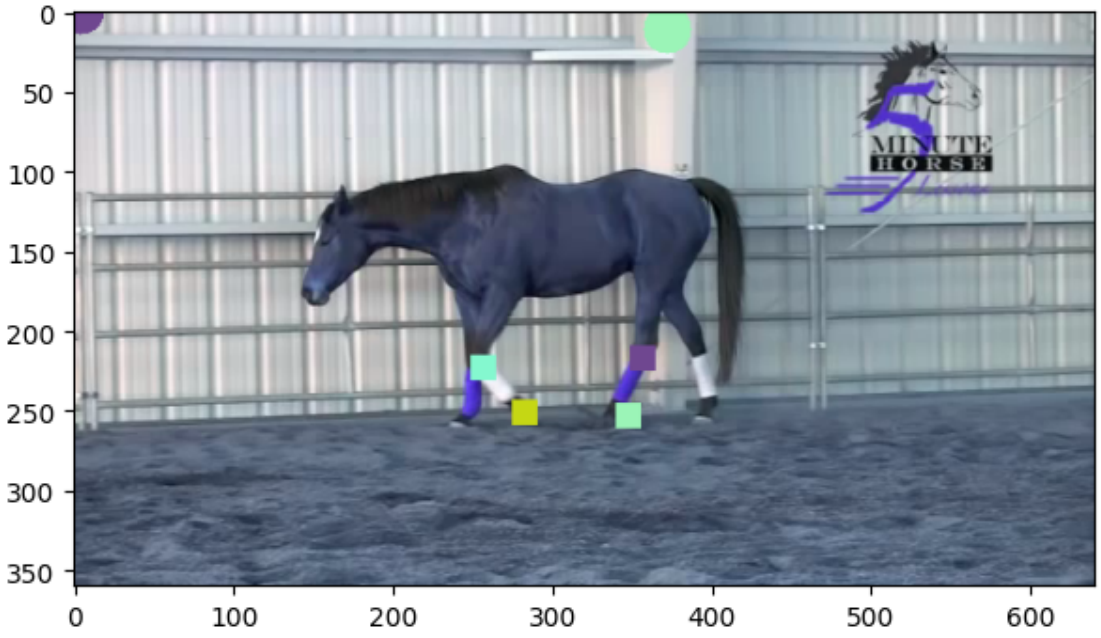

```
Epoch 1010: Loss (2646.736083984375)
```

<Figure size 640x480 with 0 Axes>

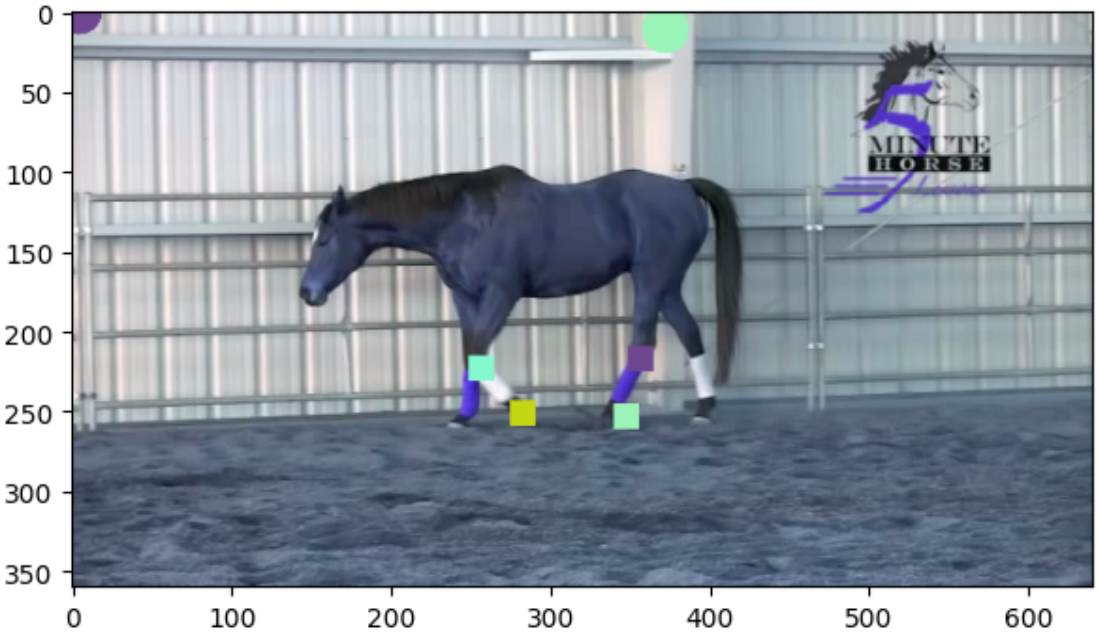


Epoch 1020: Loss (2645.074462890625)

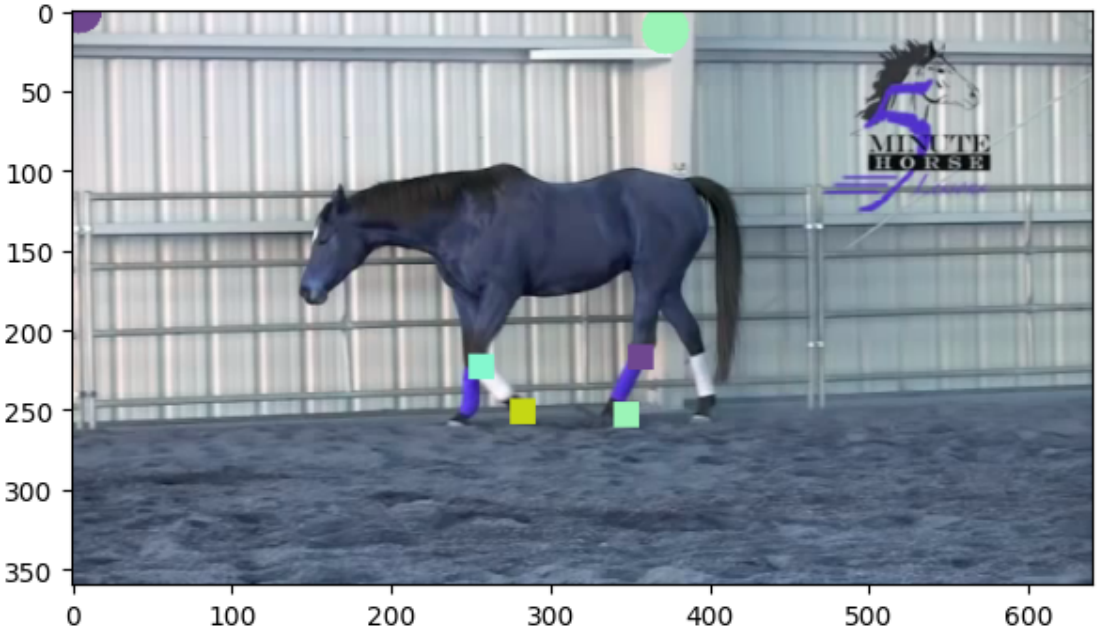
<Figure size 640x480 with 0 Axes>



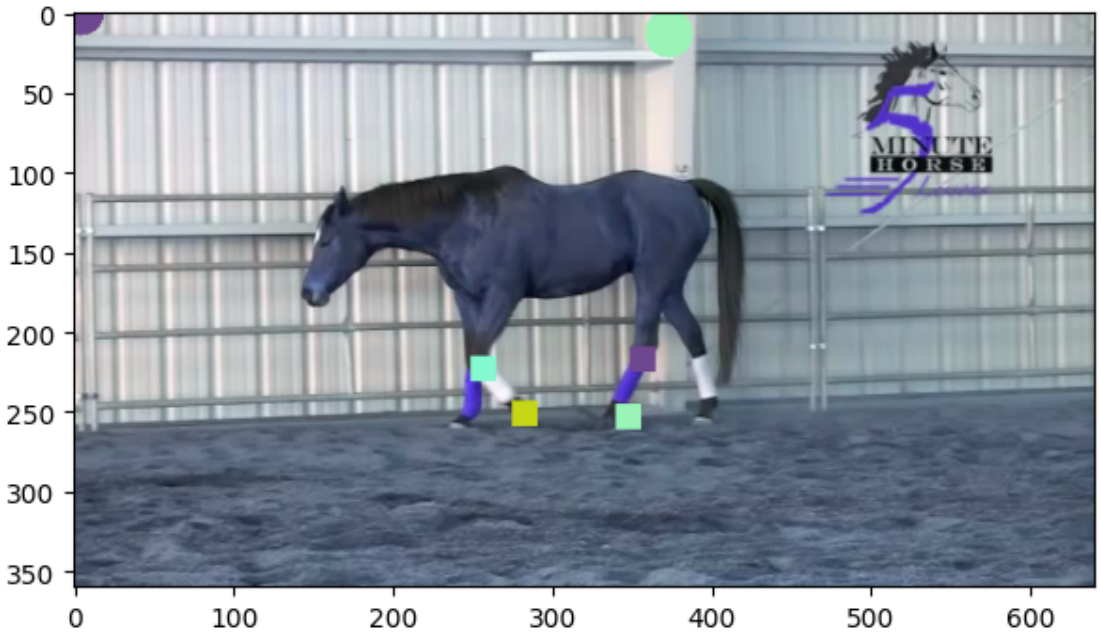
```
Epoch 1030: Loss (2643.41455078125)
<Figure size 640x480 with 0 Axes>
```



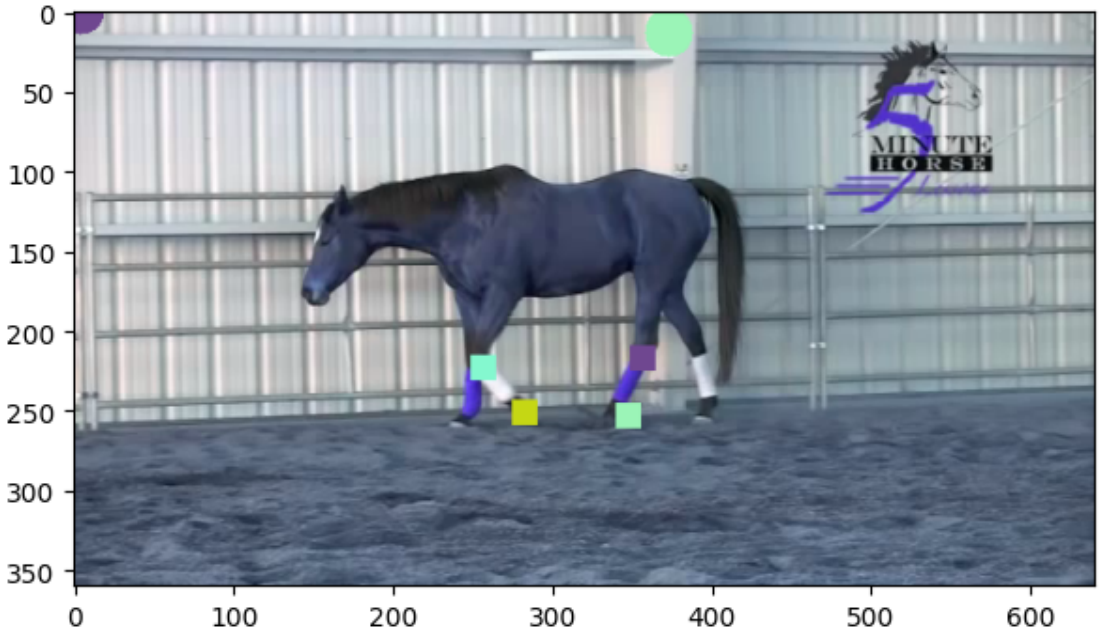

```
Epoch 1040: Loss (2641.75537109375)
<Figure size 640x480 with 0 Axes>
```



```
Epoch 1050: Loss (2640.09765625)
<Figure size 640x480 with 0 Axes>
```

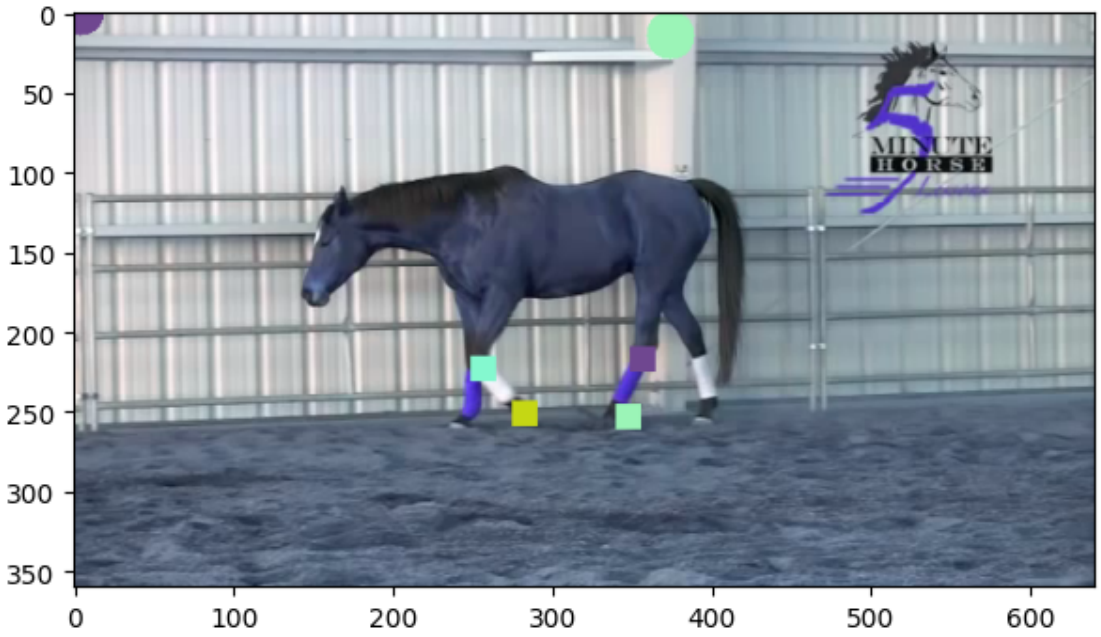



```
Epoch 1060: Loss (2638.440185546875)
<Figure size 640x480 with 0 Axes>
```



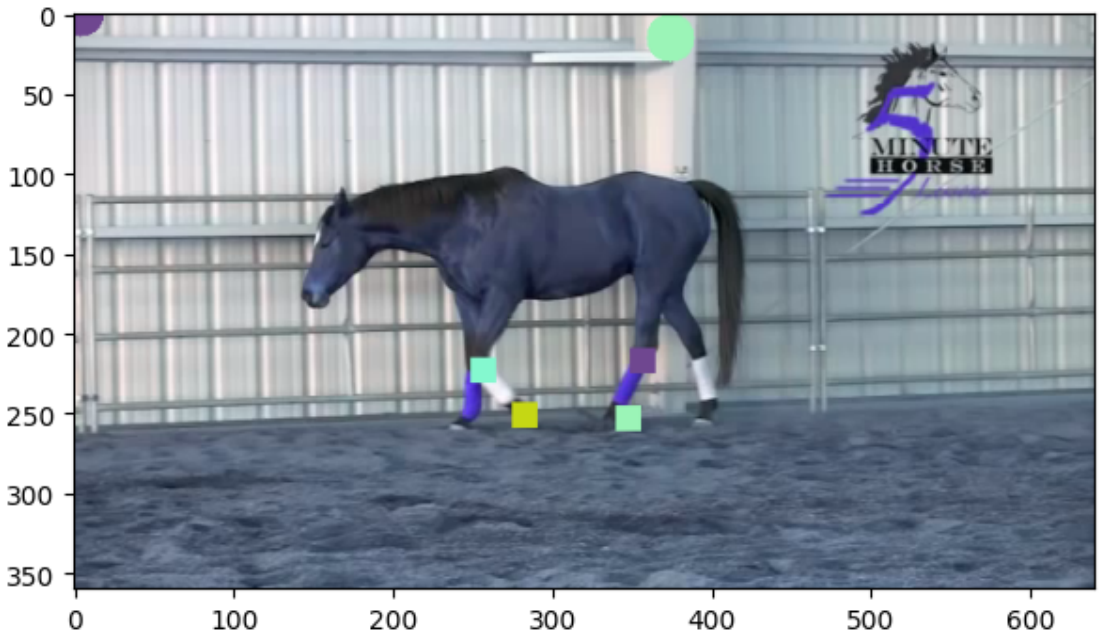
Epoch 1070: Loss (2636.782958984375)

<Figure size 640x480 with 0 Axes>

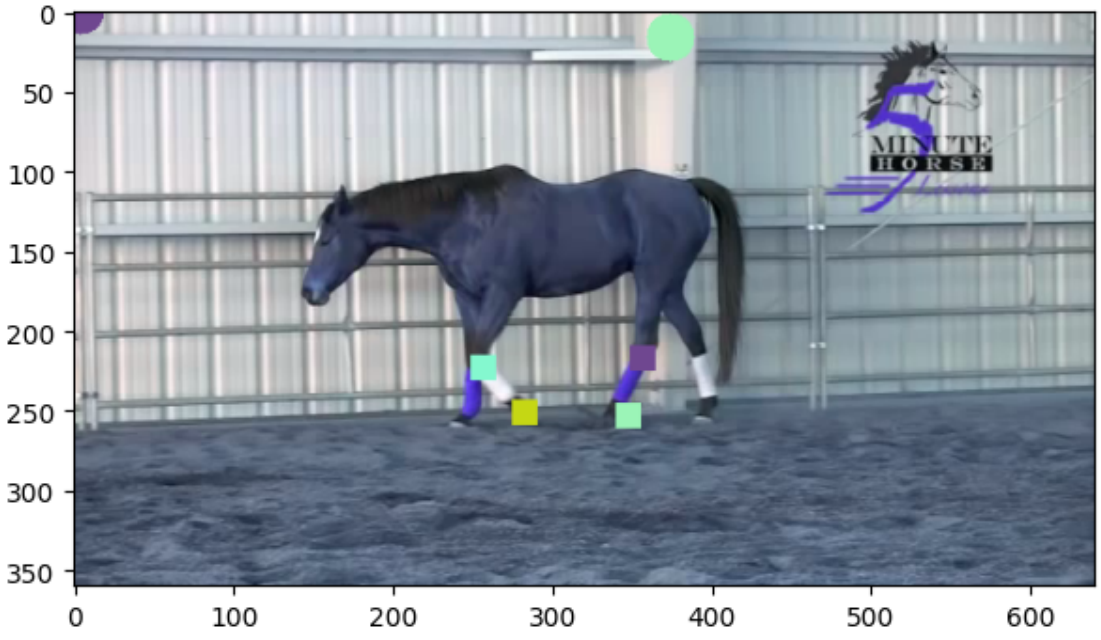


Epoch 1080: Loss (2635.125732421875)

<Figure size 640x480 with 0 Axes>

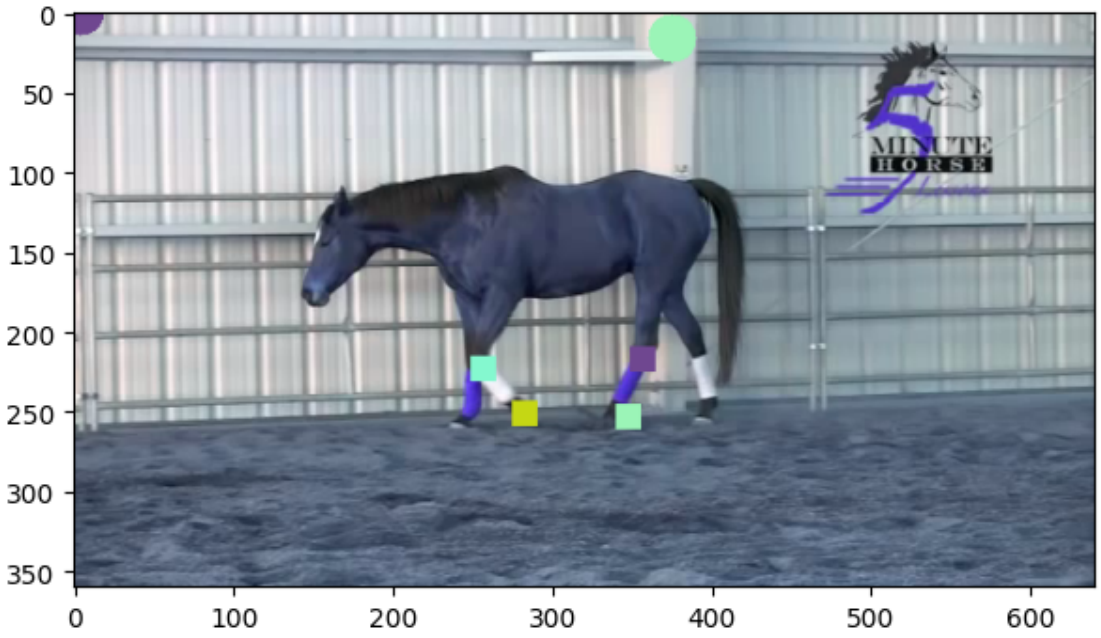



```
Epoch 1090: Loss (2633.468505859375)
<Figure size 640x480 with 0 Axes>
```



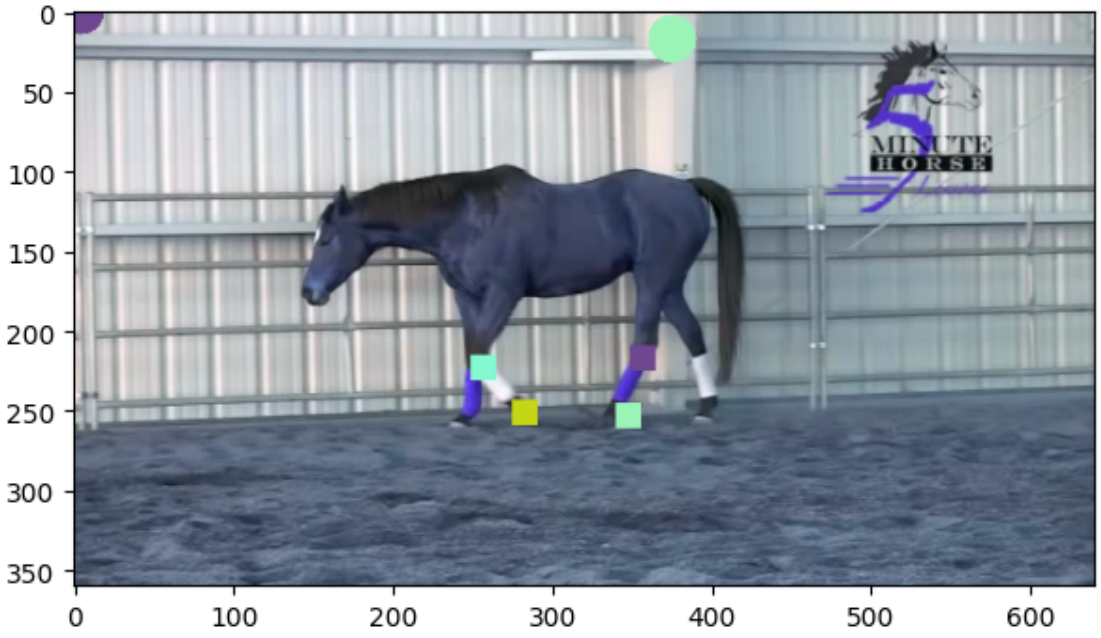

```
Epoch 1100: Loss (2631.810791015625)
```

<Figure size 640x480 with 0 Axes>



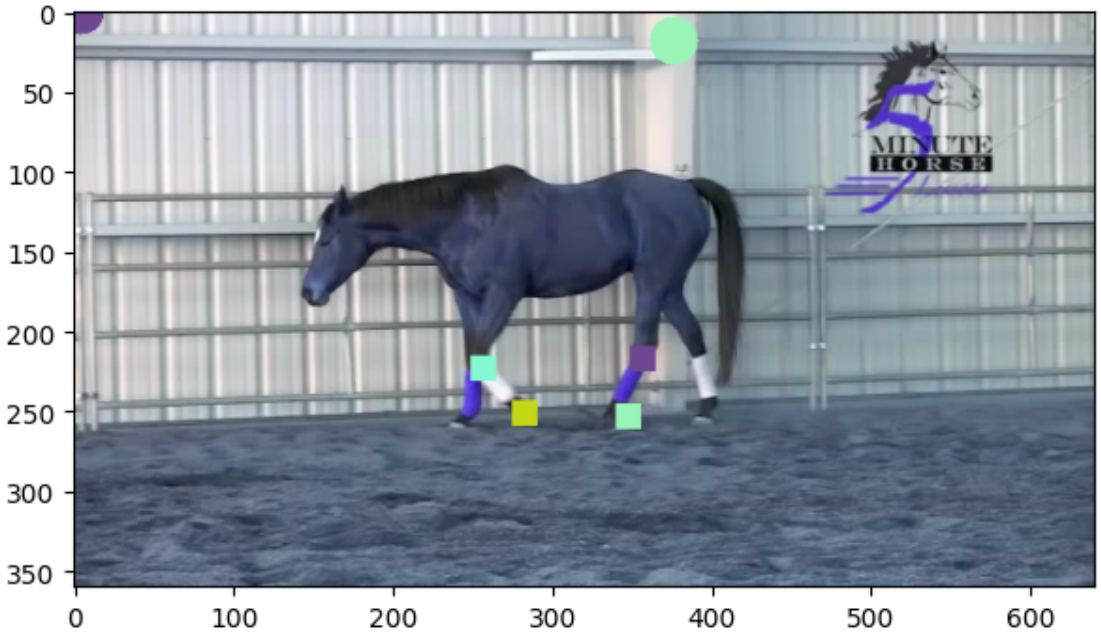
Epoch 1110: Loss (2630.152099609375)

<Figure size 640x480 with 0 Axes>



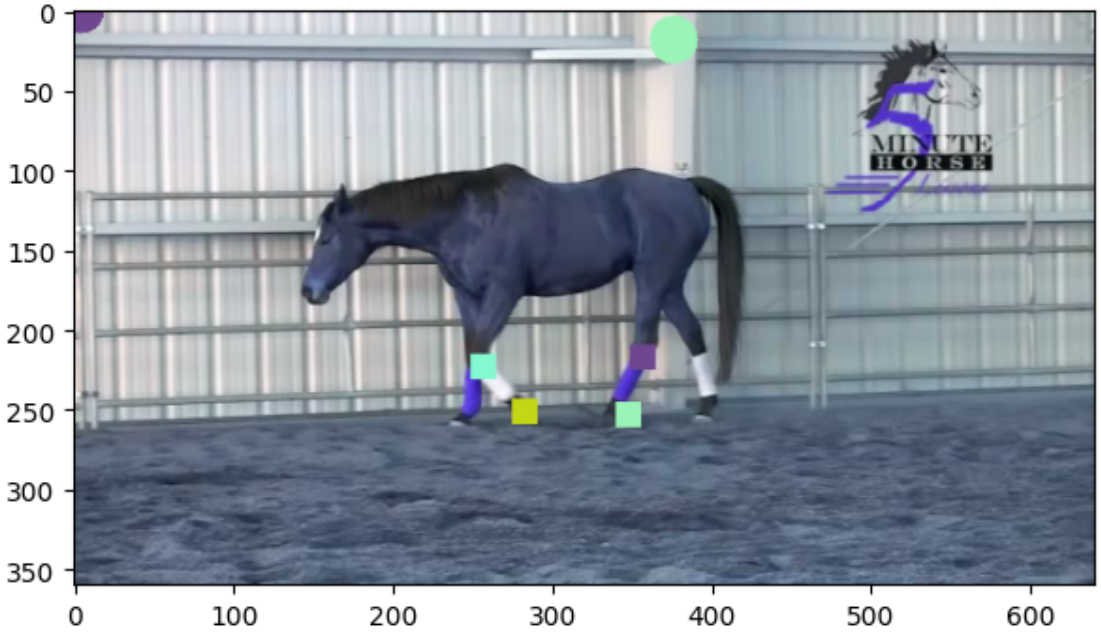
Epoch 1120: Loss (2628.492431640625)

<Figure size 640x480 with 0 Axes>



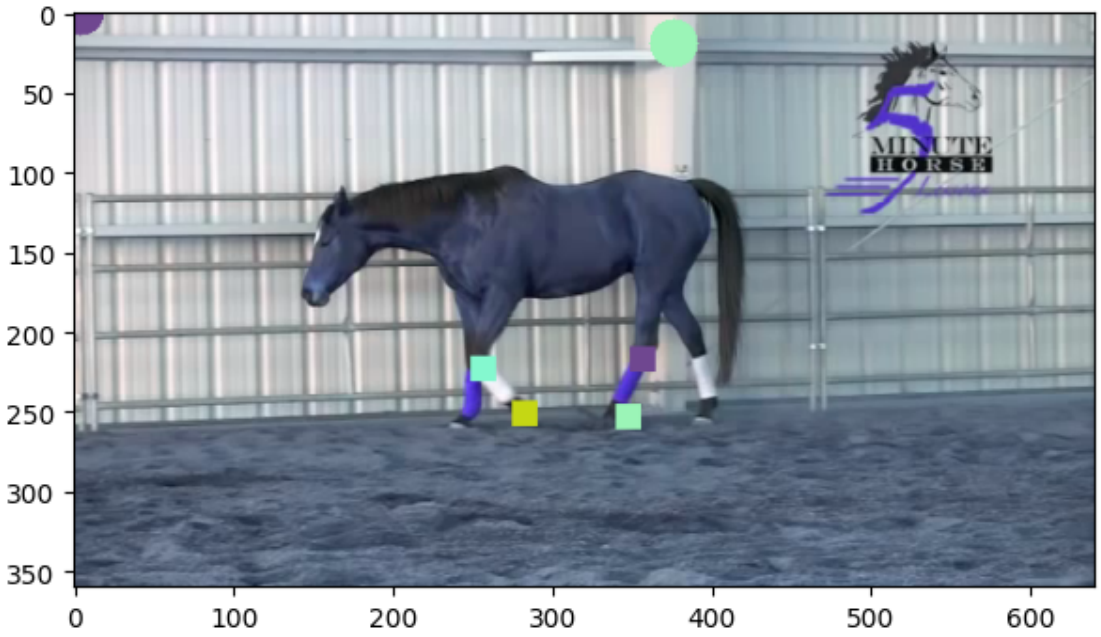
Epoch 1130: Loss (2626.831298828125)

<Figure size 640x480 with 0 Axes>



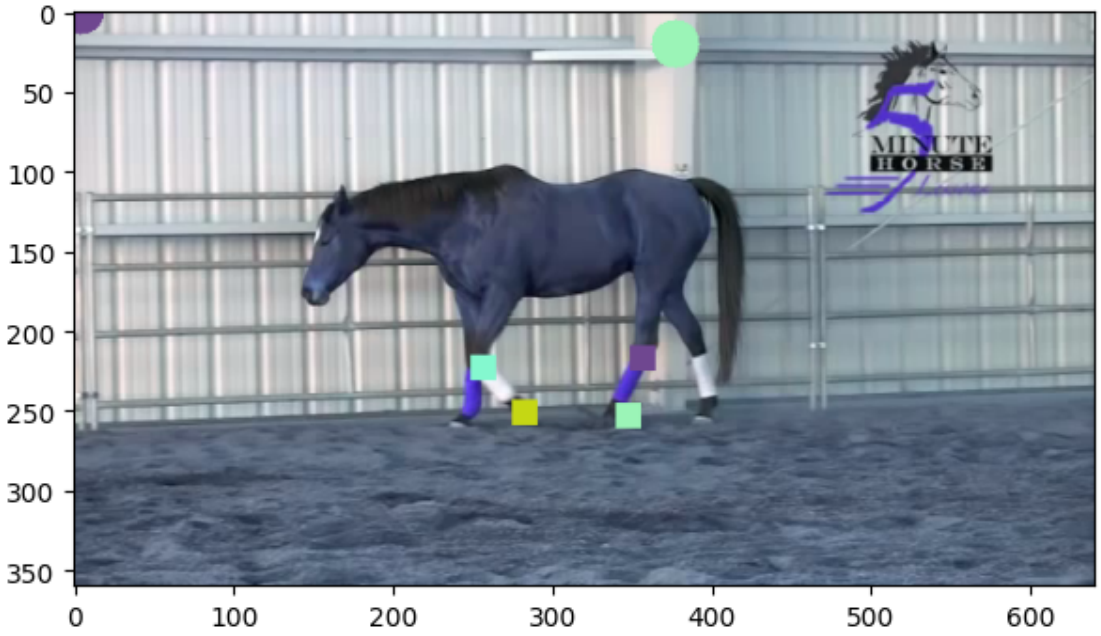
Epoch 1140: Loss (2625.168701171875)

<Figure size 640x480 with 0 Axes>



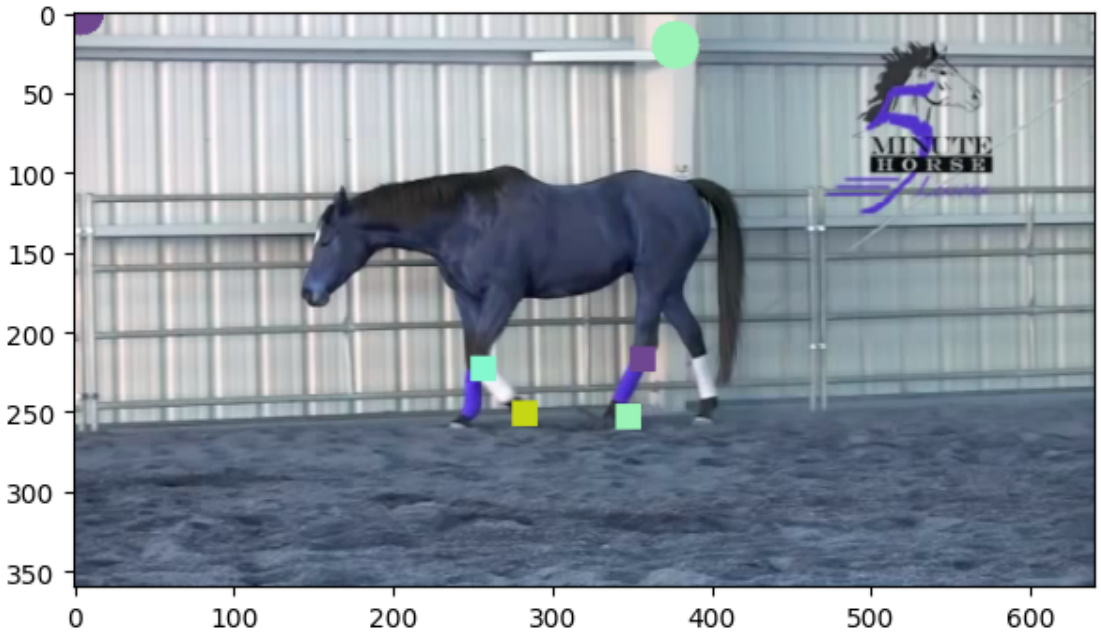
Epoch 1150: Loss (2623.50439453125)

<Figure size 640x480 with 0 Axes>



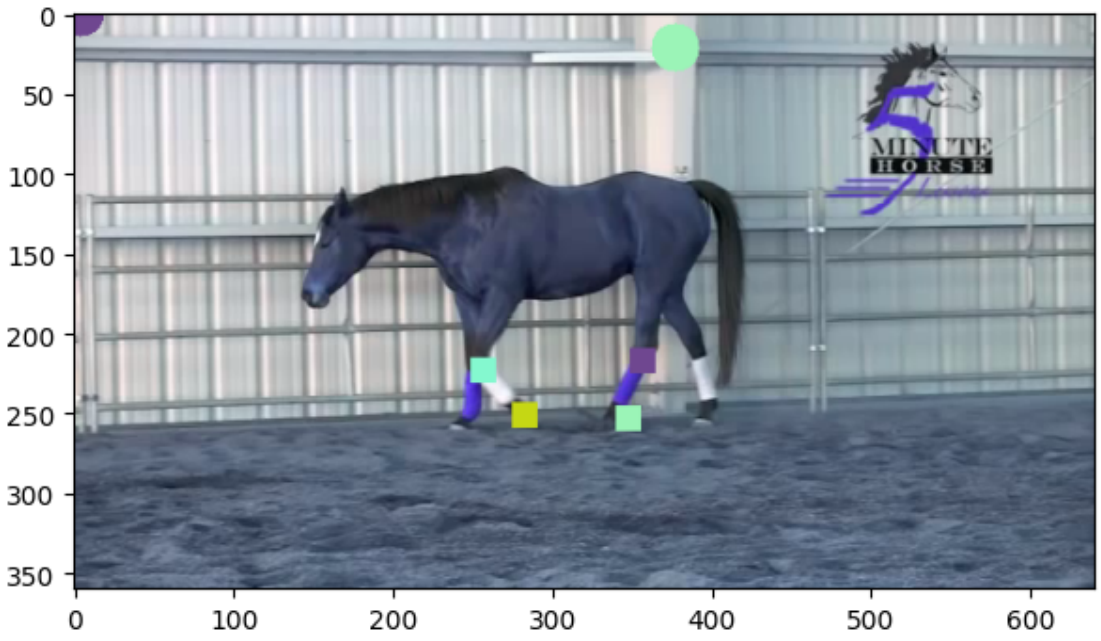
Epoch 1160: Loss (2621.837646484375)

<Figure size 640x480 with 0 Axes>



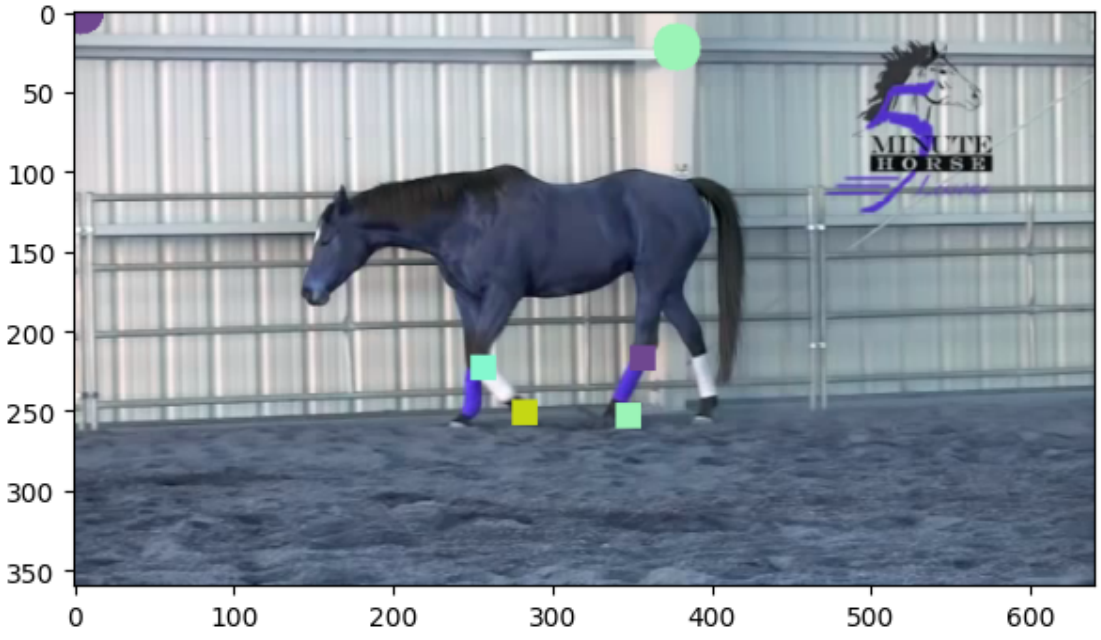
Epoch 1170: Loss (2620.16845703125)

<Figure size 640x480 with 0 Axes>



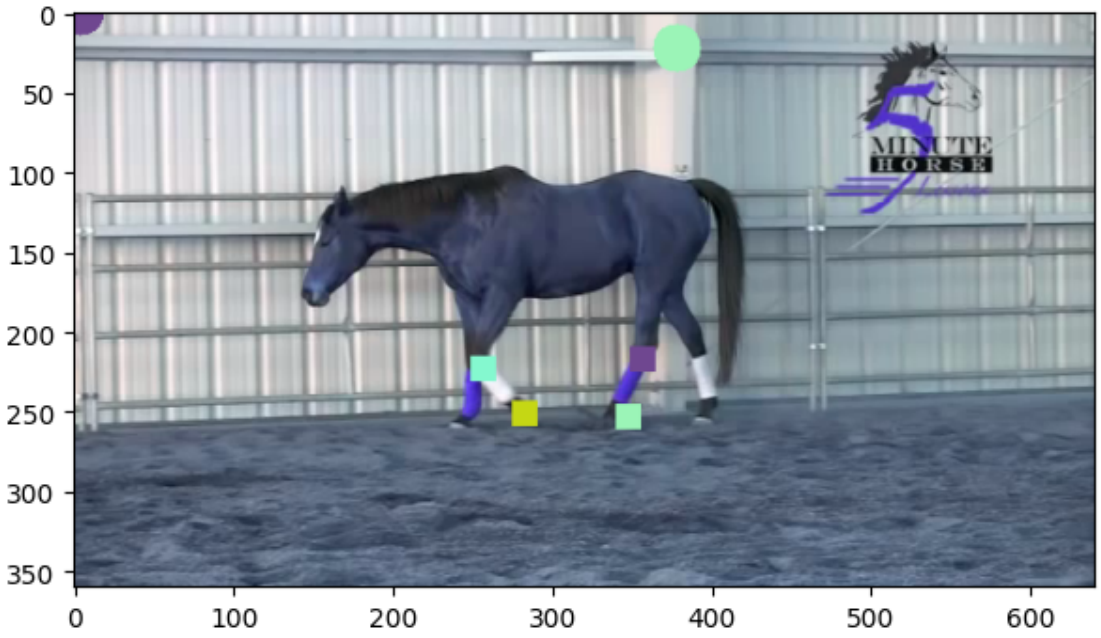
Epoch 1180: Loss (2618.496826171875)

<Figure size 640x480 with 0 Axes>



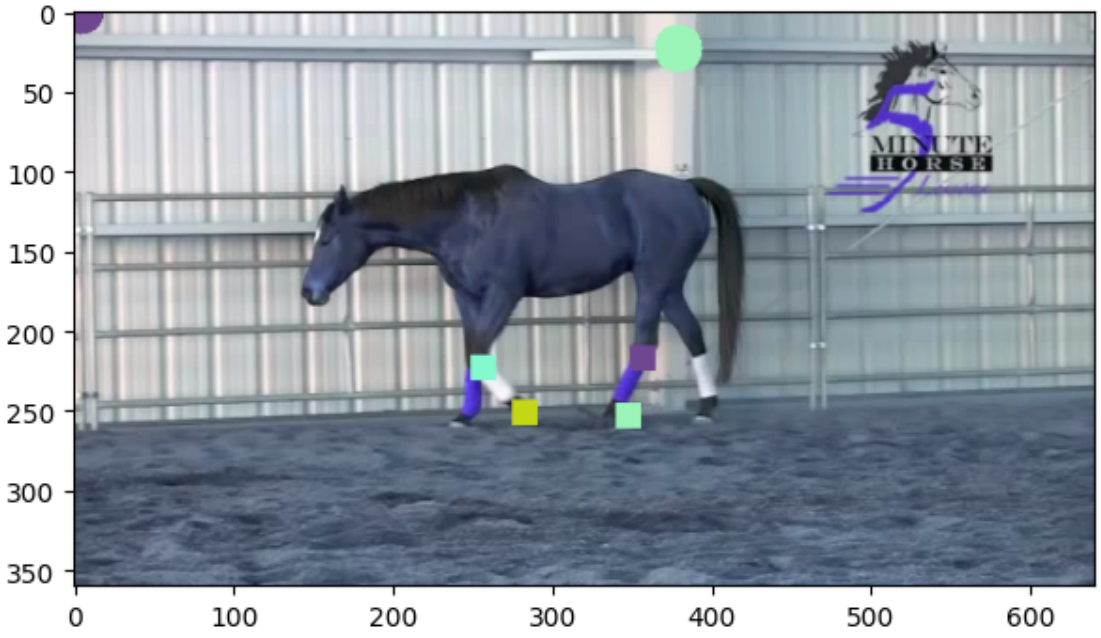
Epoch 1190: Loss (2616.822021484375)

<Figure size 640x480 with 0 Axes>



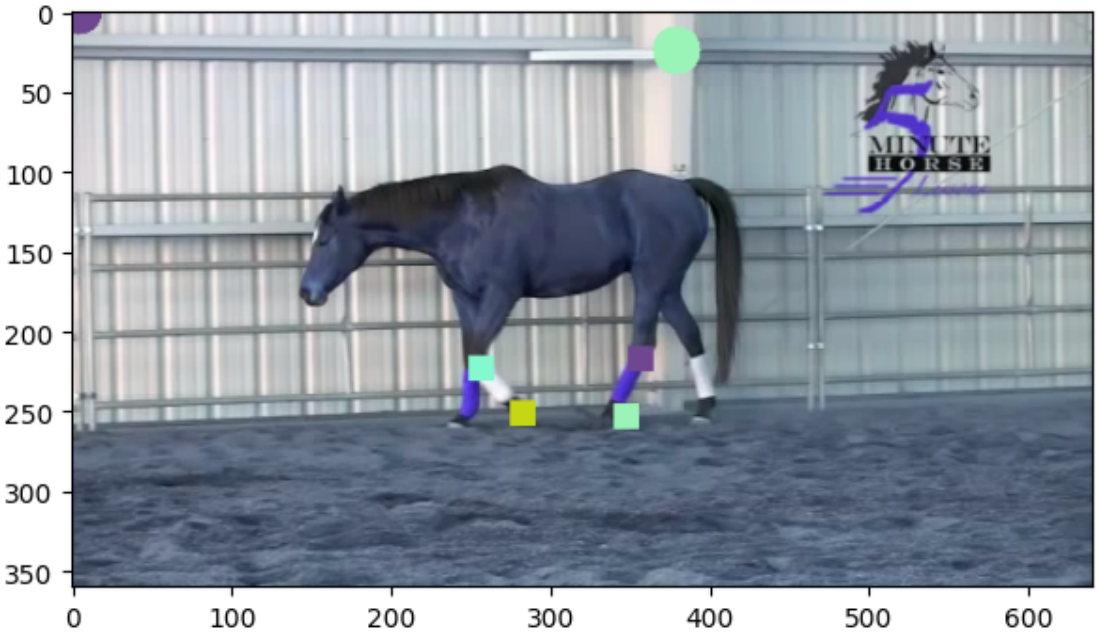
Epoch 1200: Loss (2615.14404296875)

<Figure size 640x480 with 0 Axes>



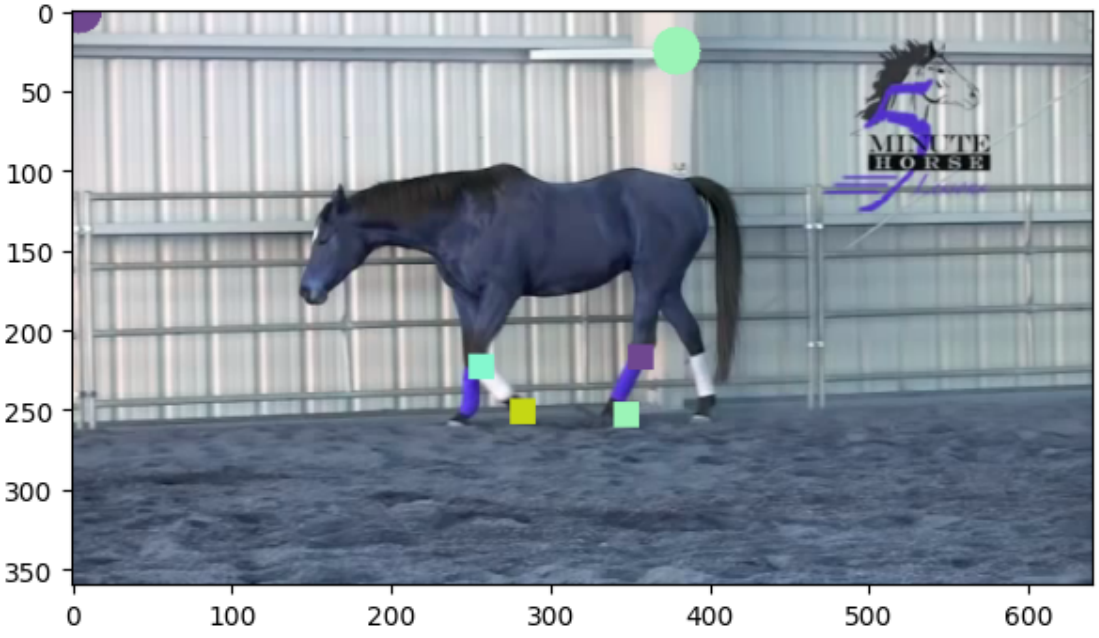
Epoch 1210: Loss (2613.46240234375)

<Figure size 640x480 with 0 Axes>



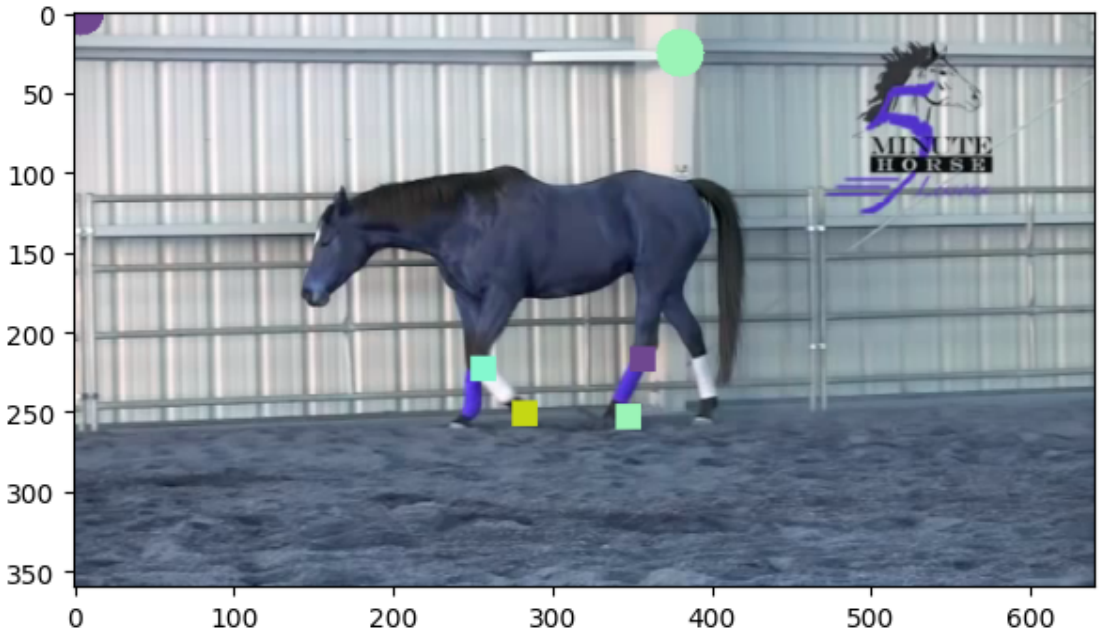
Epoch 1220: Loss (2611.77685546875)

<Figure size 640x480 with 0 Axes>



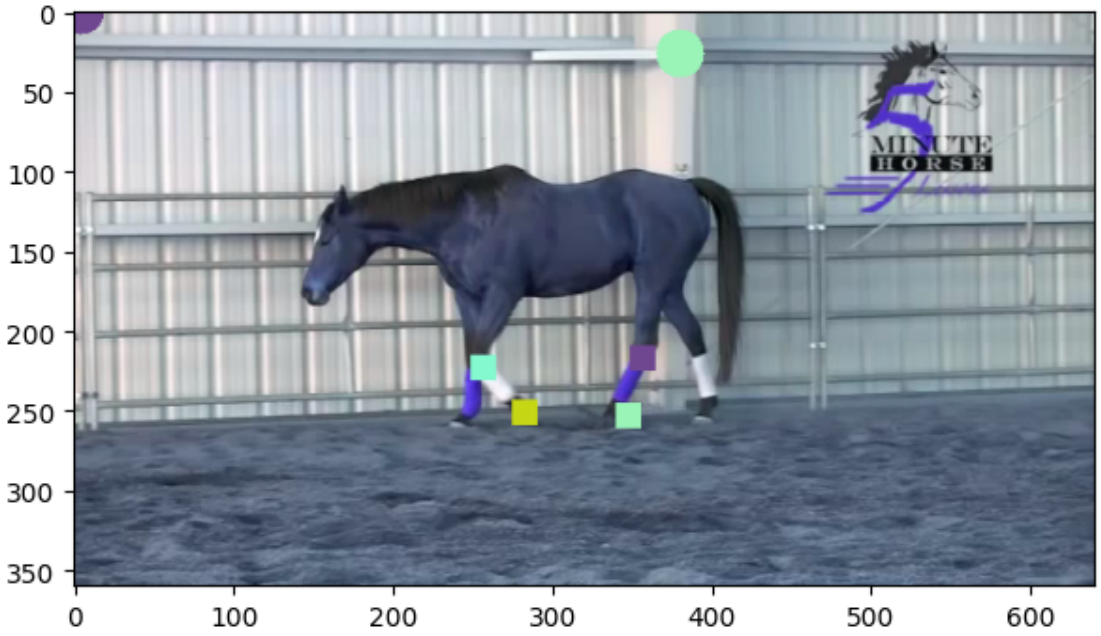
Epoch 1230: Loss (2610.087158203125)

<Figure size 640x480 with 0 Axes>



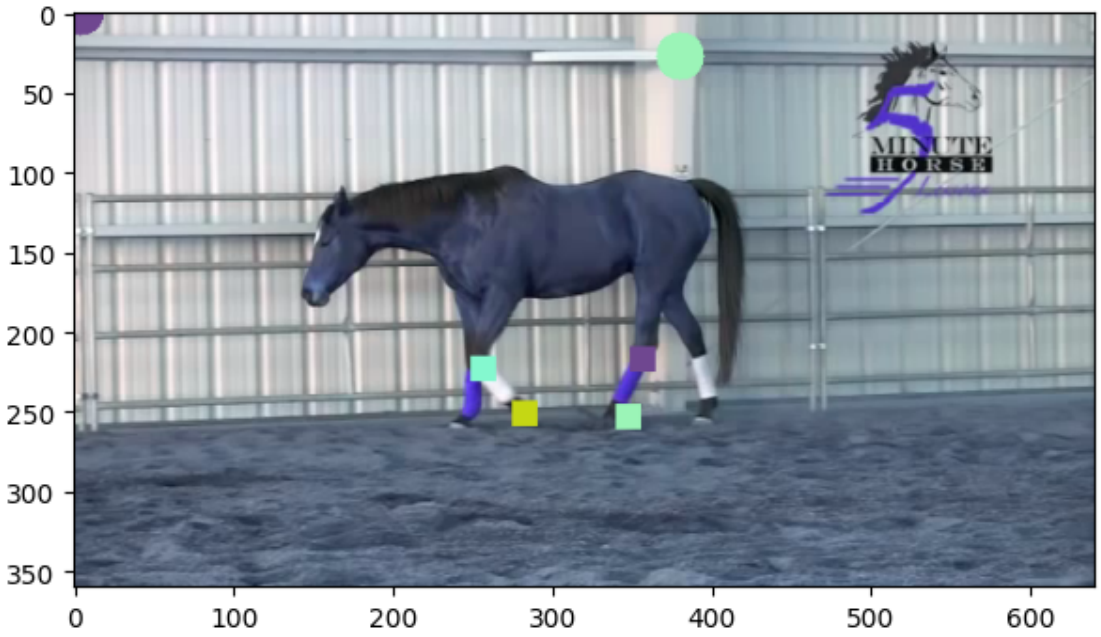
Epoch 1240: Loss (2608.393310546875)

<Figure size 640x480 with 0 Axes>



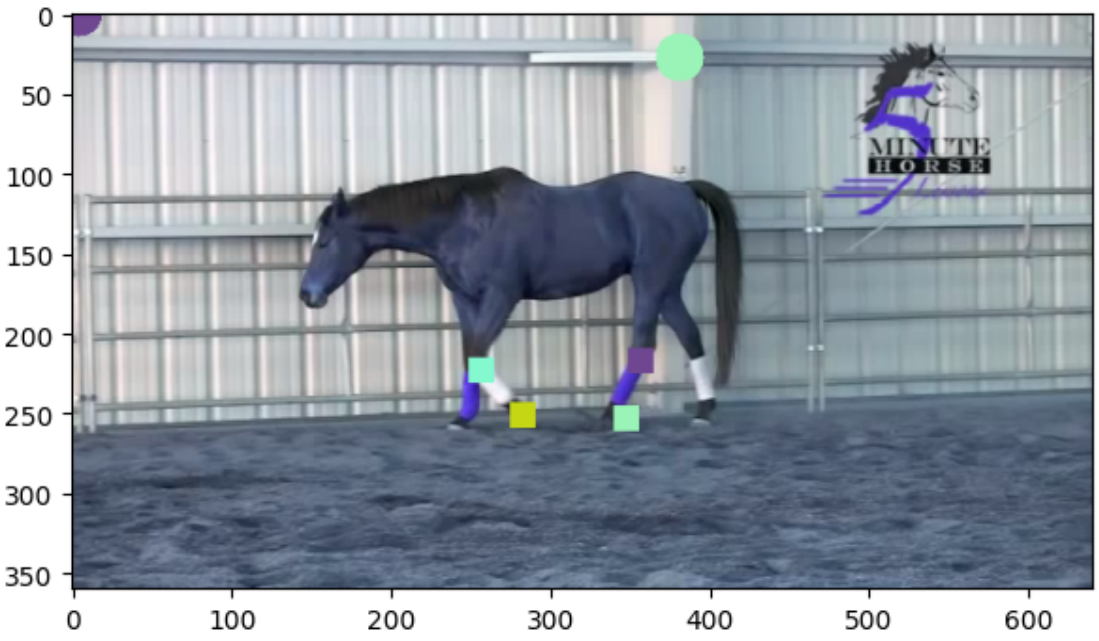
Epoch 1250: Loss (2606.694580078125)

<Figure size 640x480 with 0 Axes>

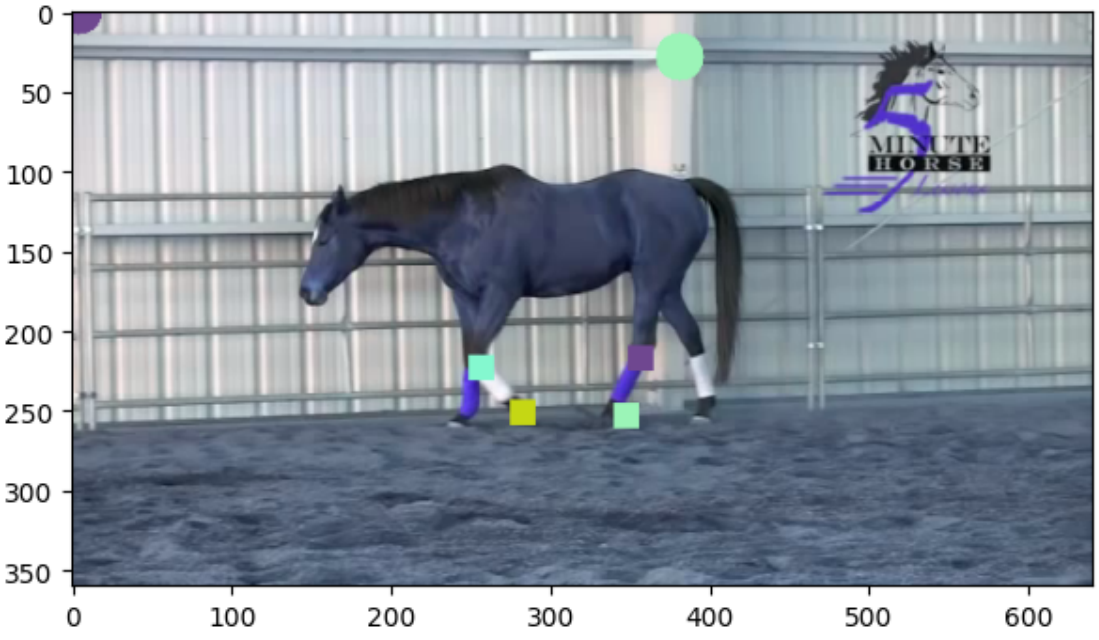


Epoch 1260: Loss (2604.990478515625)

<Figure size 640x480 with 0 Axes>

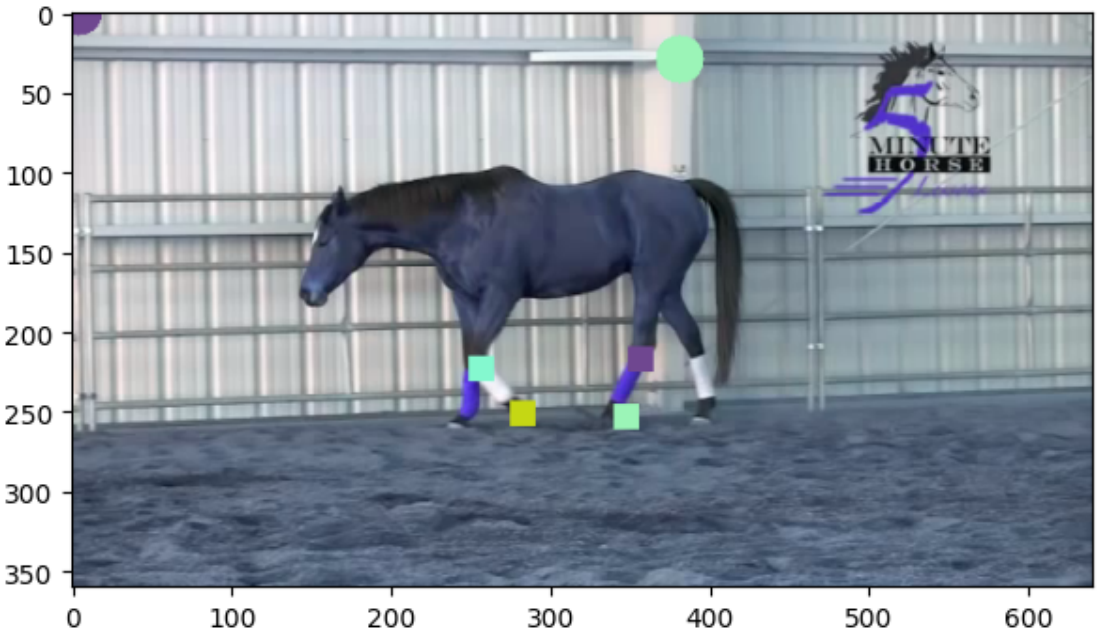



```
Epoch 1270: Loss (2603.281494140625)
<Figure size 640x480 with 0 Axes>
```



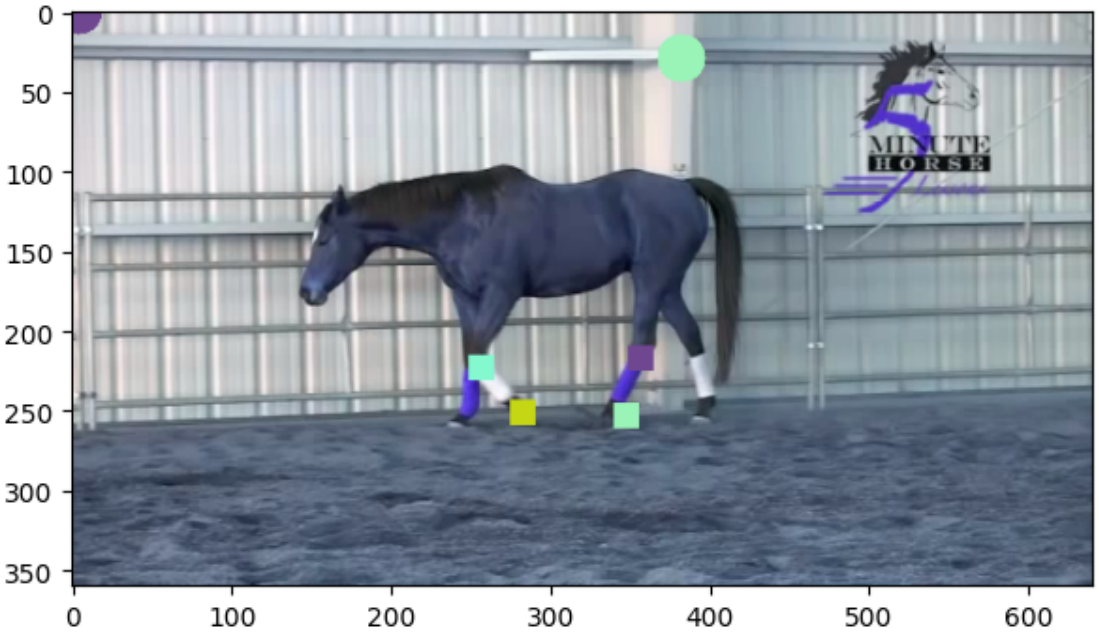

```
Epoch 1280: Loss (2601.56640625)
```

<Figure size 640x480 with 0 Axes>



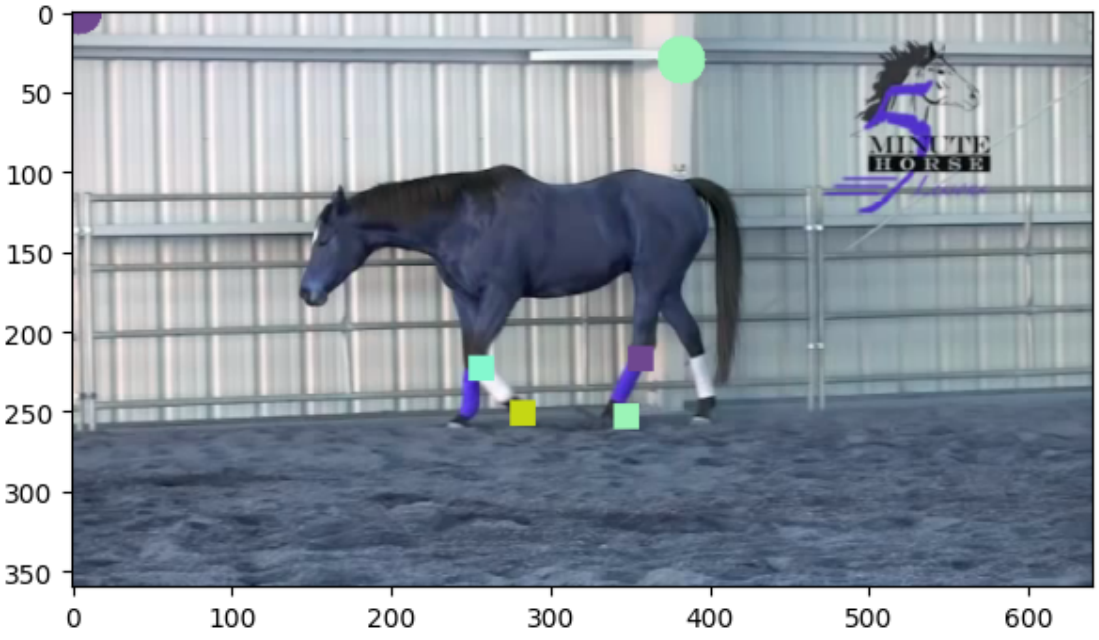
Epoch 1290: Loss (2599.845458984375)

<Figure size 640x480 with 0 Axes>



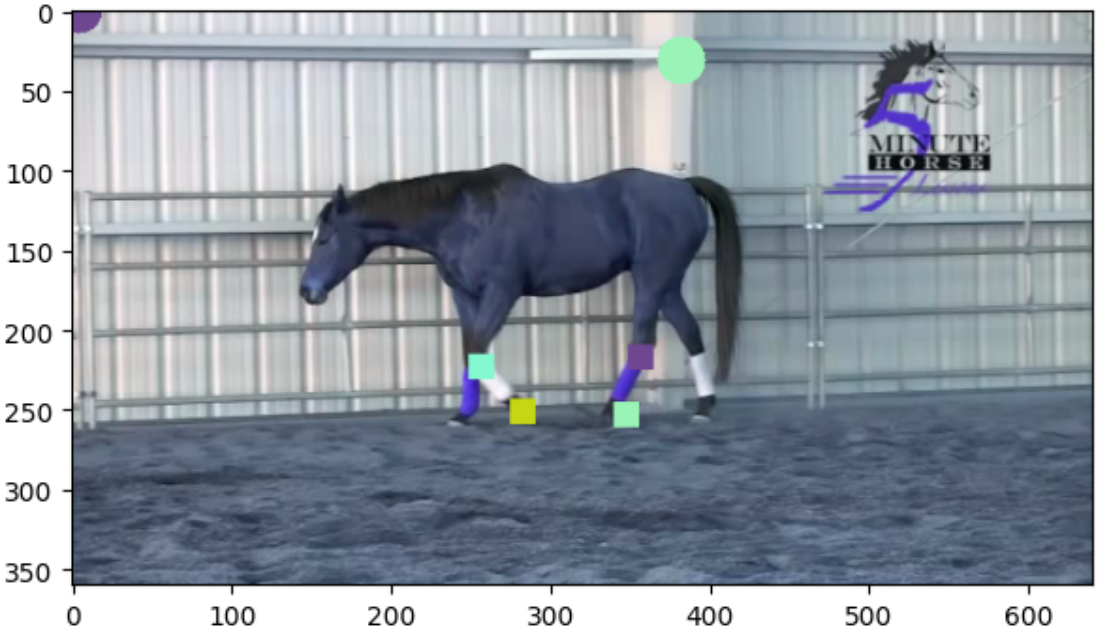
Epoch 1300: Loss (2598.1181640625)

<Figure size 640x480 with 0 Axes>

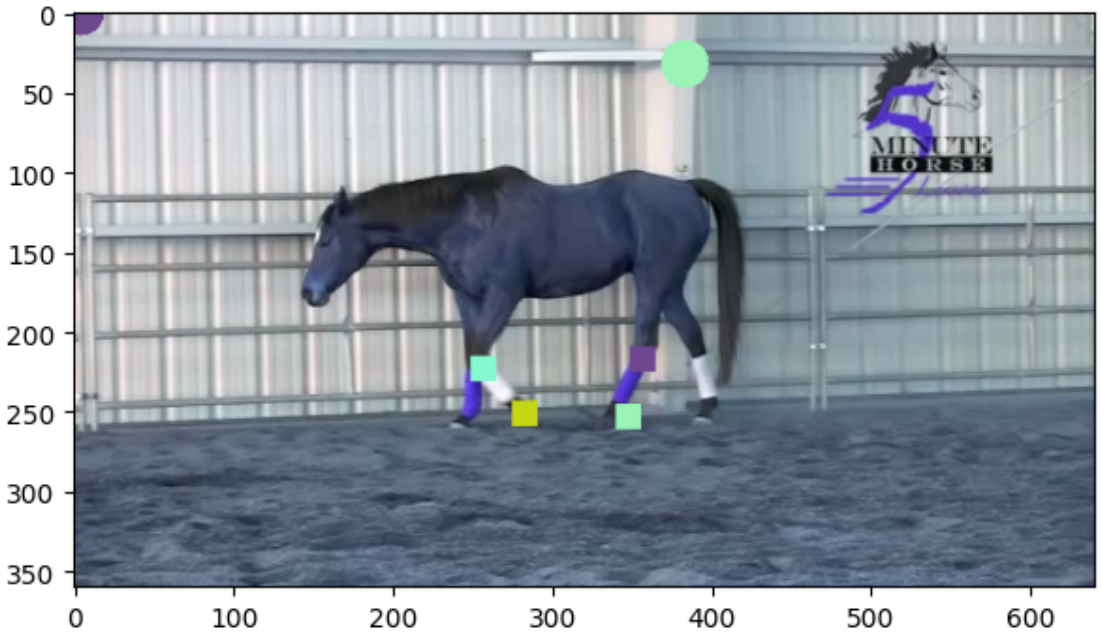


Epoch 1310: Loss (2596.384033203125)

<Figure size 640x480 with 0 Axes>



```
Epoch 1320: Loss (2594.642822265625)
<Figure size 640x480 with 0 Axes>
```




```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[37], line 14
     12 for epoch in range(2000):
     13     optimizer.zero_grad()
---> 14     cameraEstimate = getRobotPositionInCamera(joint_angles_array,␣
 ↪conversionMatrix, learnable_robot_model)
     15     debugPrint(f"Camera Estimate Shape: {cameraEstimate.shape}")
     17     #Loss Between Where Robot/Horse Should Be and Where It Is

Cell In[28], line 25, in getRobotPositionInCamera(joint_angles_array,␣
 ↪conversionMatrix, learnable_robot_model)
     24 def getRobotPositionInCamera(joint_angles_array, conversionMatrix,␣
 ↪learnable_robot_model):
---> 25     return getRobotPositionsInPixelSpace(conversionMatrix,␣
 ↪getRobotPositionsFromJointAngles(joint_angles_array, learnable_robot_model))

Cell In[28], line 11, in getRobotPositionsFromJointAngles(joint_angles_array,␣
 ↪learnable_robot_model)
      7 #Where the Robot Thinks It Is
      8 for joint_angles in joint_angles_array:
```

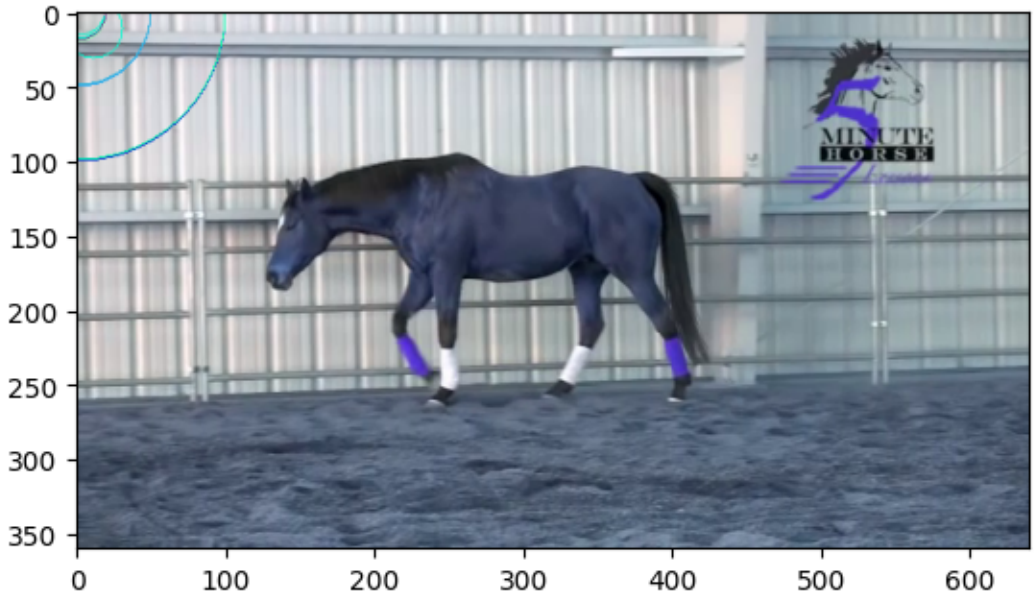
```
      9     projection = torch.cat((learnable_robot_model.
 ↪compute_forward_kinematics(joint_angles, "FL_foot")[0],\
     10     learnable_robot_model.compute_forward_kinematics(joint_angles,␣
 ↪"FL_calf")[0],\
---> 11     learnable_robot_model.compute_forward_kinematics(joint_angles,␣
 ↪"RR_foot")[0],\
     12     learnable_robot_model.compute_forward_kinematics(joint_angles,␣
 ↪"RR_calf")[0]))
     13     projection = projection.unsqueeze(0).unsqueeze(0)
     14     projections.append(projection)

File ~/Library/Mobile Documents/com~apple~CloudDocs/Work/UCSD/Research - Yip/
 ↪differentiable-robot-model/differentiable_robot_model/robot_model.py:74, in␣
 ↪tensor_check.<locals>.wrapper(self, *args, **kwargs)
     69 processed_kwargs = {
     70     key: preprocess(kwargs[key], self, batch_info) for key in kwargs
     71 }
     73 # Perform function
---> 74 ret = function(self, *processed_args, **processed_kwargs)
     76 # Parse output
     77 if type(ret) is torch.Tensor:

File ~/Library/Mobile Documents/com~apple~CloudDocs/Work/UCSD/Research - Yip/
 ↪differentiable-robot-model/differentiable_robot_model/robot_model.py:243, in␣
 ↪DifferentiableRobotModel.compute_forward_kinematics(self, q, link_name,␣
 ↪recursive)
    241 else:
    242     qd = torch.zeros_like(q)
--> 243     self.update_kinematic_state(q, qd)
    245     pose = self._bodies[self._name_to_idx_map[link_name]].pose
    246     pos = pose.translation()

File ~/Library/Mobile Documents/com~apple~CloudDocs/Work/UCSD/Research - Yip/
 ↪differentiable-robot-model/differentiable_robot_model/robot_model.py:74, in␣
 ↪tensor_check.<locals>.wrapper(self, *args, **kwargs)
     69 processed_kwargs = {
     70     key: preprocess(kwargs[key], self, batch_info) for key in kwargs
     71 }
     73 # Perform function
---> 74 ret = function(self, *processed_args, **processed_kwargs)
     76 # Parse output
     77 if type(ret) is torch.Tensor:

File ~/Library/Mobile Documents/com~apple~CloudDocs/Work/UCSD/Research - Yip/
 ↪differentiable-robot-model/differentiable_robot_model/robot_model.py:193, in␣
 ↪DifferentiableRobotModel.update_kinematic_state(self, q, qd)
    189     new_vel = parent_body.vel.transform(parentToChildT)
    191     # this body's angular velocity is combination of the velocity␣
 ↪experienced at it's parent's link
```

```
    192     # + the velocity created by this body's joint
--> 193     body.vel = body.joint_vel.add_motion_vec(new_vel)
    195 return

File ~/opt/anaconda3/envs/DEEPLABCUT/lib/python3.8/site-packages/torch/nn/
 ↪modules/module.py:1255, in Module.__setattr__(self, name, value)
   1253     buffers[name] = value
   1254 else:
-> 1255     object.__setattr__(self, name, value)

KeyboardInterrupt:
```

[ ]:
```
#Plot Projected End Effector Positions onto Horse Images
```

[ ]:
```
import matplotlib.pyplot as plt
from random import randrange
imgIndex = 2
img = images[imgIndex]
for x,y in getRobotPositionInCamera(joint_angles_array, conversionMatrix,␣
 ↪learnable_robot_model)[imgIndex][0]:
    color = (randrange(0,256),randrange(0,256),randrange(0,256))
    img = cv2.circle(img, (int(x), int(y)), 100, color)
plt.imshow(img)
```

[ ]:
```
<matplotlib.image.AxesImage at 0x7f97239b7c70>
```



[ ]:
```
training_data[0]
```

[ ]:
```
tensor([[275.799, 244.884],
        [249.626, 216.098],
        [340.429, 246.946],
        [349.002, 210.681]])
```

[ ]:
```

```