data_loader_verifaction

December 13, 2024

[1]:
```
### Data Loader Function:

# Group Activity Data Loader:
#         1. Can return a full image of target frame with its group label␣
 ↪(frame, tensor(8)) *needed for B1*.
#         2. Can return a all player crops of the target frame with its group␣
 ↪label all player have same label  ( (12, crop frame), tensor(1,8)) *needed␣
 ↪for B3 step B, C*.
#         3. Can return a full clip with each frame dir with its group label␣
 ↪(all the same) ((9, frame) , tensor(9,8)) *needed for B4*.
#   4. Can return a full clip with all player crop with its group label (all␣
 ↪the same) ((12, 9, crop frame), tensor(9,8)) *needed for B5, B6, B7*.

#  Person Activity Data Loader:
#         1. Can return crop of player image frames in independent way (crop␣
 ↪frame , tensor(9)) *needed for B3 step A , B6*.
#         2. Can return crop of player in the same clip (12 , 9, crop frame) ,␣
 ↪(tensor(12, 9, 9)) *needed for B5, B7*.

# Note:
# 1.  Frame and crop frame means all image dim (C, H, W).
# 2.  The Sort flag (sort the player by the tracer id) *needed for B8*.

##########################################################################################
import os
import sys
import torch
import matplotlib.pyplot as plt
import torchvision.transforms as T
import albumentations as A
from albumentations.pytorch import ToTensorV2
from torchvision.transforms import v2

PROJECT_ROOT= "/teamspace/studios/this_studio/Group-Activity-Recognition"
sys.path.append(os.path.abspath(PROJECT_ROOT))
from data_loader import Person_Activity_DataSet, Group_Activity_DataSet
```

```
dataset_root = "/teamspace/studios/this_studio/Group-Activity-Recognition/data"
annot_path =   f"{dataset_root}/annot_all.pkl"
videos_path =  f"{dataset_root}/videos"

people_activity_clases = ["Waiting", "Setting", "Digging", "Falling",
 ↪"Spiking"        , "Blocking", "Jumping"        , "Moving", "Standing"]
person_activity_labels  = {class_name.lower():i for i, class_name in
 ↪enumerate(people_activity_clases)}

group_activity_clases = ["r_set", "r_spike" , "r-pass", "r_winpoint",
 ↪"l_winpoint", "l-pass", "l-spike", "l_set"]
group_activity_labels  = {class_name:i for i, class_name in
 ↪enumerate(group_activity_clases)}

train_spilt = [1, 3, 6, 7, 10, 13, 15, 16, 18, 22, 23, 31, 32, 36, 38, 39, 40,
 ↪41, 42, 48, 50, 52, 53, 54]
```

0.0.1 Test people activity data loader

1. Can return crop of player image frames in independent way (crop frame , tensor(9)) ***needed for B3 step A , B6.***

[2]:
```
transforms = A.Compose([
    A.Resize(224, 224),
    ToTensorV2()
])

data_loader = Person_Activity_DataSet(videos_path, annot_path,
 ↪split=train_spilt, seq=False, labels=person_activity_labels,
 ↪transform=transforms)
```

[3]:
```
len(data_loader)
```

[3]: 231327

[4]:
```
frame , label = data_loader[0]

label.shape # (,9) class of person activity
```

[4]: torch.Size([9])

[5]:
```
label
```

[5]: tensor([0., 0., 0., 0., 0., 0., 0., 0., 1.], dtype=torch.float64)

[6]:
```
frame.shape # (C, H, W)
```

[6]:
```
torch.Size([3, 224, 224])
```

[7]:
```
frame , label = data_loader[50]

label_idex = label.argmax().item()
print(f"{people_activity_clases[label_idex]}")

plt.figure(figsize=(2, 2))
plt.imshow(frame.permute(1,2,0))  # Converts from (C, H, W) to (H, W, C)
plt.axis('off')  # Optional: to hide axes
plt.show()
```

```
Standing
```

[8]:
```
frame , label = data_loader[450]

label_idex = label.argmax().item()
print(f"{people_activity_clases[label_idex]}")

plt.figure(figsize=(2, 2))
plt.imshow(frame.permute(1,2,0))  # Converts from (C, H, W) to (H, W, C)
plt.axis('off')  # Optional: to hide axes
plt.show()
```

```
Setting
```

[9]:
```
frame , label = data_loader[120]

label_idex = label.argmax().item()
print(f"{people_activity_clases[label_idex]}")

plt.figure(figsize=(2, 2))
plt.imshow(frame.permute(1,2,0))  # Converts from (C, H, W) to (H, W, C)
plt.axis('off')  # Optional: to hide axes
plt.show()
```

```
Standing
```

[10]:
```
frame , label = data_loader[800]

label_idex = label.argmax().item()
print(f"{people_activity_clases[label_idex]}")

plt.figure(figsize=(2, 2))
plt.imshow(frame.permute(1,2,0))  # Converts from (C, H, W) to (H, W, C)
plt.axis('off')  # Optional: to hide axes
plt.show()
```

```
Falling
```

2. Can return crop of player in the same clip (12 , 9, crop frame) , (tensor(12, 9, 9)) *needed for B5, B7.*

[11]:
```
transforms = A.Compose([
    A.Resize(224, 224),
    ToTensorV2()
])

data_loader = Person_Activity_DataSet(videos_path, annot_path,␣
 ↪split=train_spilt, seq=True, labels=person_activity_labels,␣
 ↪transform=transforms)
```

[12]:
```
len(data_loader)
```

[12]: 2152

[13]:
```
clip, label = data_loader[100]

label.shape # (12 player , 9 frame , label of 9 class)
```

[13]: torch.Size([12, 9, 9])

[14]:
```
label[0, 0, :]
```

[14]: tensor([0., 0., 0., 0., 0., 0., 0., 0., 1.], dtype=torch.float64)

[15]:
```
clip.shape #  (12 player, 9 frame, C, H, W)
```

[15]: torch.Size([12, 9, 3, 224, 224])

[16]:
```
label_idex = label[0, 0].argmax().item()
print(f"{people_activity_clases[label_idex]}")

plt.figure(figsize=(2, 2)) # frist player  - first frame
plt.imshow(clip[0, 0].permute(1, 2, 0).numpy())  # Converts from (C, H, W) to␣
 ↪(H, W, C)
plt.axis('off')  # Optional: to hide axes
plt.show()
```

Standing

[17]:
```
label_idex = label[0, 2].argmax().item()
print(f"{people_activity_clases[label_idex]}")

plt.figure(figsize=(2, 2)) # frist player  - Thrid frame
plt.imshow(clip[0, 2].permute(1, 2, 0).numpy())  # Converts from (C, H, W) to␣
↪(H, W, C)
plt.axis('off')
plt.show()
```

```
Standing
```

[18]:
```
label_idex = label[0, 4].argmax().item()
print(f"{people_activity_clases[label_idex]}")

plt.figure(figsize=(2, 2))
plt.imshow(clip[0, 4].permute(1, 2, 0).numpy())  # Converts from (C, H, W) to␣
↪(H, W, C)
plt.axis('off')
plt.show()
```

```
Standing
```

[19]:
```
label_idex = label[0, 6].argmax().item()
print(f"{people_activity_clases[label_idex]}")

plt.figure(figsize=(2, 2))
plt.imshow(clip[0, 6].permute(1, 2, 0).numpy())  # Converts from (C, H, W) to␣
↪(H, W, C)
plt.axis('off')
plt.show()
```

```
Standing
```

[20]:
```
label_idex = label[0, 8].argmax().item()
print(f"{people_activity_clases[label_idex]}")

plt.figure(figsize=(2, 2)) # frist player  - last frame
plt.imshow(clip[0, 8].permute(1, 2, 0).numpy())  # Converts from (C, H, W) to␣
↪(H, W, C)
plt.axis('off')
plt.show()
```

```
Standing
```

[21]:
```
label_idex = label[8, 2].argmax().item()
print(f"{people_activity_clases[label_idex]}")

plt.figure(figsize=(2, 2))
plt.imshow(clip[1, 0].permute(1, 2, 0).numpy())  # Converts from (C, H, W) to␣
↪(H, W, C)
plt.axis('off')
plt.show()
```

```
Standing
```

[22]:
```
label_idex = label[8, 4].argmax().item()
print(f"{people_activity_clases[label_idex]}")

plt.figure(figsize=(2, 2))
plt.imshow(clip[1, 4].permute(1, 2, 0).numpy())  # Converts from (C, H, W) to␣
↪(H, W, C)
plt.axis('off')
plt.show()
```

```
Standing
```

[23]:
```
label_idex = label[8, 6].argmax().item()
print(f"{people_activity_clases[label_idex]}")

plt.figure(figsize=(2, 2))
plt.imshow(clip[1, 6].permute(1, 2, 0).numpy())  # Converts from (C, H, W) to␣
↪(H, W, C)
plt.axis('off')
plt.show()
```

```
Standing
```

[24]:
```
label_idex = label[8, 8].argmax().item()
print(f"{people_activity_clases[label_idex]}")

plt.figure(figsize=(2, 2)) # second player  - last frame
plt.imshow(clip[1, 8].permute(1, 2, 0).numpy())  # Converts from (C, H, W) to␣
↪(H, W, C)
plt.axis('off')
plt.show()
```

```
Standing
```

0.1 Test Group Activity data loader

1. Can return a full image of target frame with its group label (frame, tensor(8)) *needed for B1.*

[25]:
```
transforms = A.Compose([
    A.Resize(224, 224),
    ToTensorV2()
])


data_loader = Group_Activity_DataSet(videos_path, annot_path,␣
 ↪split=train_spilt, crops=False , seq=False, labels=group_activity_labels,␣
 ↪transform=transforms)
```

[26]:
```
len(data_loader)
```

[26]: 19368

[27]:
```
frame , label = data_loader[0]
```

[28]:
```
print(label.shape) # (,8)
label
```

```
torch.Size([8])
```

[28]: tensor([0., 0., 0., 0., 0., 1., 0., 0.])

[29]:
```
frame.shape # (C, H , W)
```

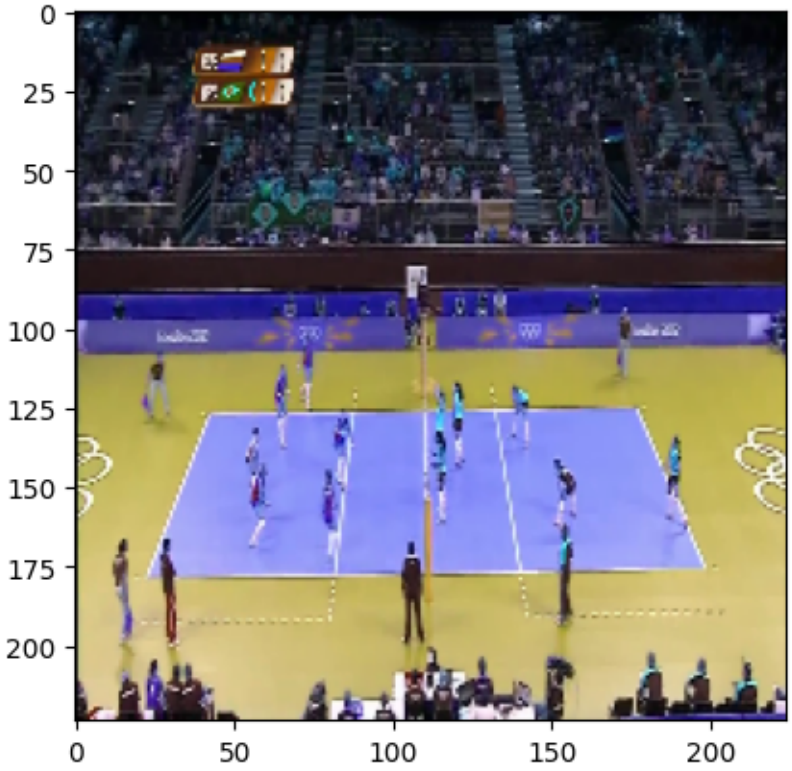
[29]: torch.Size([3, 224, 224])

[30]:
```
index = data_loader[0][1].argmax().item()
print(f"{group_activity_clases[index]}")

plt.imshow(data_loader[0][0].permute(1,2,0))
```

```
plt.show()
```

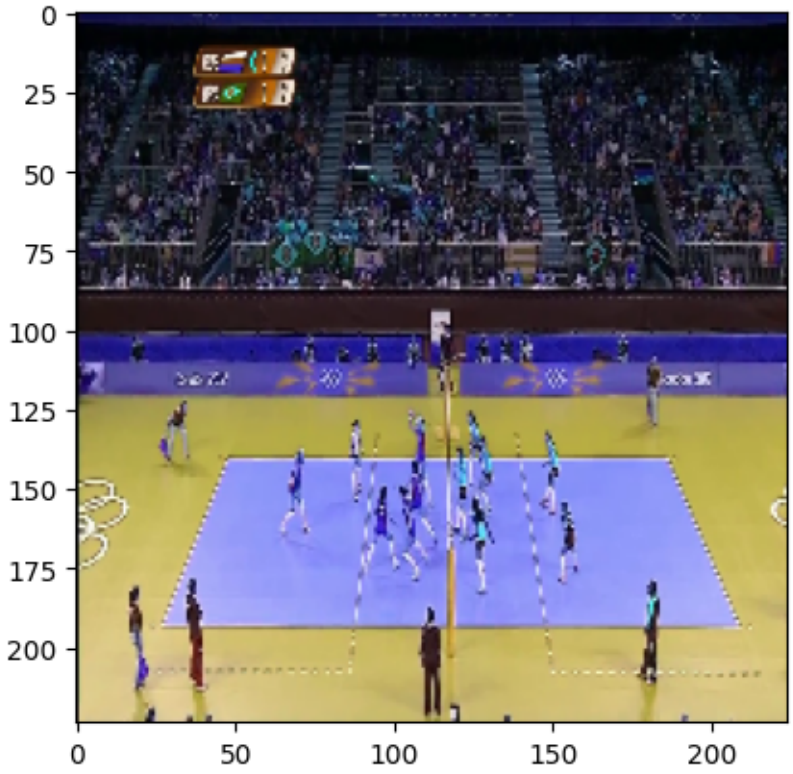
```
l-pass
```



[31]:
```
index = data_loader[152][1].argmax().item()
print(f"{group_activity_clases[index]}")

plt.imshow(data_loader[152][0].permute(1,2,0))
plt.show()
```

```
l-pass
```



2. Can return a all player crops of the target frame with its group label (all player have same label) ((12, crop frame), tensor(1,8)) *needed for B3 step B, C.*

[32]:
```
transforms = A.Compose([
    A.Resize(224, 224),
    ToTensorV2()
])

data_loader= Group_Activity_DataSet(videos_path, annot_path, split=train_spilt,␣
 ↪crops=True , seq=False, labels=group_activity_labels, transform=transforms)
```

[33]:
```
len(data_loader) # the differents between case 1 and 2 the input consist of 12␣
 ↪bbox
```

[33]: 19368

[34]:
```
frame_crops, label = data_loader[152]
```

[35]:
```
print(label.shape) # (,8)
label
```

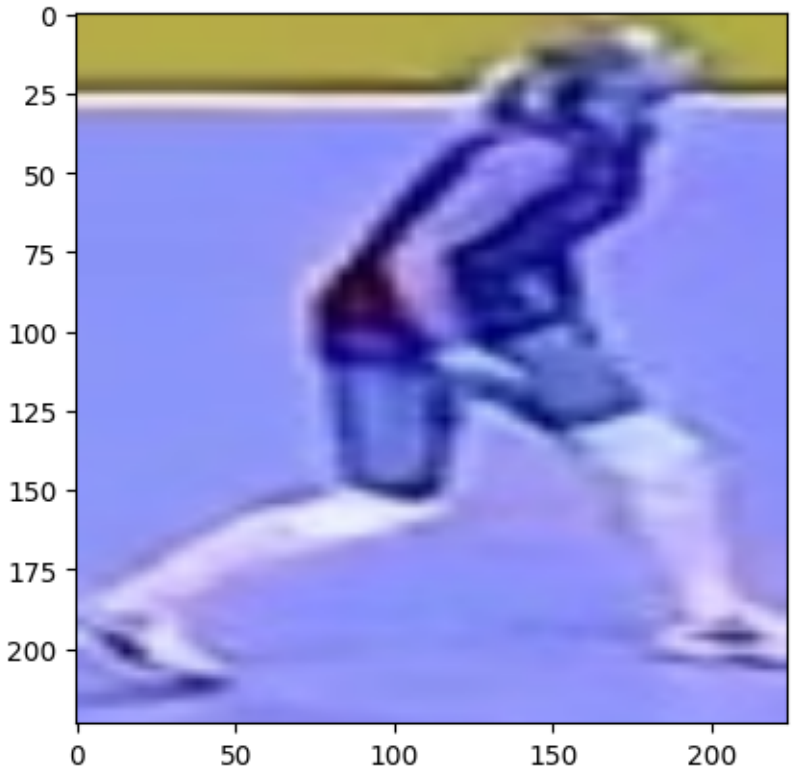
```
torch.Size([8])
```

[35]:
```
tensor([0., 0., 0., 0., 0., 1., 0., 0.])
```

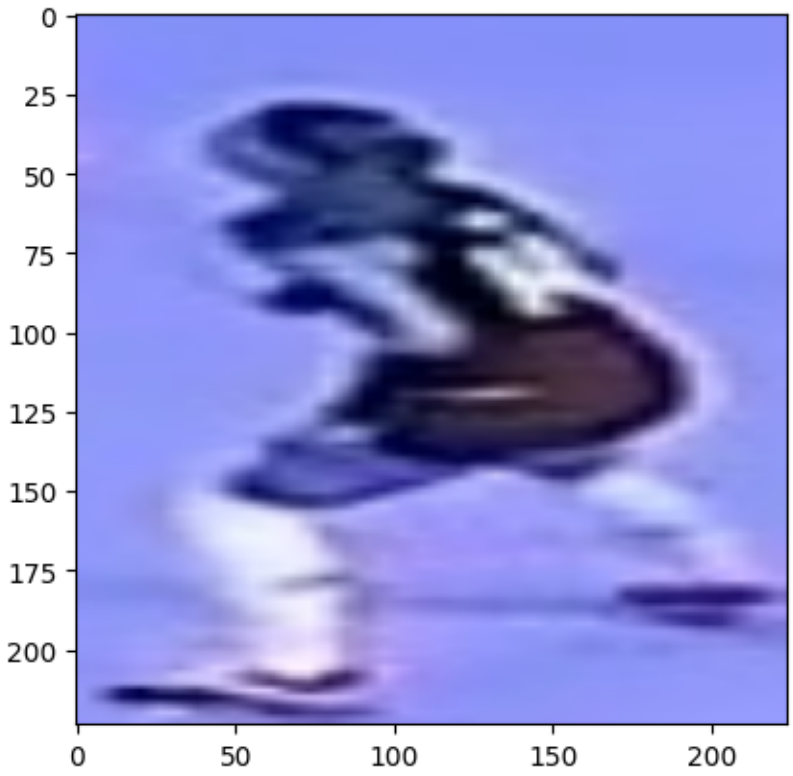
[36]:
```
frame_crops.shape # (12, C, H, W) ---> 12 bbox of the frame
```

[36]:
```
torch.Size([12, 3, 224, 224])
```

[37]:
```
plt.imshow(frame_crops[0].permute(1,2,0))  # first bbox of the fraem
plt.show()
```



[38]:
```
plt.imshow(frame_crops[11].permute(1,2,0))  # last bbox of the frame
plt.show()
```



3. Can return a full clip with each frame dir with its group label (all the same) ((9, frame) , tensor(9,8)) *needed for B4*.

[39]:
```
transforms = A.Compose([
    A.Resize(224, 224),
    ToTensorV2()
])

data_loader = Group_Activity_DataSet(videos_path, annot_path,␣
 ↪split=train_spilt, crops=False , seq=True, labels=group_activity_labels,␣
 ↪transform=transforms)
```

[40]:
```
len(data_loader)
```

[40]:
```
2152
```

[41]:
```
clip , label = data_loader[100]
clip.shape # (9 frames, C, H, W)
```

[41]:
```
torch.Size([9, 3, 224, 224])
```

[42]:
```
group_activity_labels
```

[42]:
```
{'r_set': 0,
 'r_spike': 1,
 'r-pass': 2,
 'r_winpoint': 3,
 'l_winpoint': 4,
 'l-pass': 5,
 'l-spike': 6,
 'l_set': 7}
```

[43]:
```
plt.figure(figsize=(2, 2)) # First frame of the clip
plt.imshow(clip[0].permute(1, 2, 0).numpy())
plt.axis('off')
plt.show()
```

[44]:
```
plt.figure(figsize=(2, 2)) # second frame of the clip
plt.imshow(clip[1].permute(1, 2, 0).numpy())
plt.axis('off')
plt.show()
```

[45]:
```
plt.figure(figsize=(2, 2))
plt.imshow(clip[2].permute(1, 2, 0).numpy())
plt.axis('off')
```

```
plt.show()
```

[46]:
```
plt.figure(figsize=(2, 2))
plt.imshow(clip[3].permute(1, 2, 0).numpy())
plt.axis('off')
plt.show()
```

[47]:
```
plt.figure(figsize=(2, 2))
plt.imshow(clip[4].permute(1, 2, 0).numpy())
plt.axis('off')
plt.show()
```

[48]:
```
plt.figure(figsize=(2, 2))
plt.imshow(clip[5].permute(1, 2, 0).numpy())
plt.axis('off')
plt.show()
```

[49]:
```
plt.figure(figsize=(2, 2))
plt.imshow(clip[6].permute(1, 2, 0).numpy())
plt.axis('off')
plt.show()
```

[50]:
```
plt.figure(figsize=(2, 2))
plt.imshow(clip[7].permute(1, 2, 0).numpy())
plt.axis('off')
plt.show()
```

[51]:
```
plt.figure(figsize=(2, 2)) # Last frame of the clip
plt.imshow(clip[8].permute(1, 2, 0).numpy())
plt.axis('off')
plt.show()
```

4. Can return a full clip with all player crop with its group label (all the same) ((12, 9, crop frame), tensor(9,8)) *needed for B5, B6, B7.*

[52]:
```
transforms = A.Compose([
    A.Resize(224, 224),
    ToTensorV2()
])

data_loader = Group_Activity_DataSet(videos_path, annot_path,␣
 ↪split=train_spilt, crops=True , seq=True, labels=group_activity_labels,␣
 ↪transform=transforms)
```

[53]:
```
len(data_loader)
```

[53]: 2152

[54]:
```
clip, label = data_loader[50]
```

[55]:
```
label.shape # (9, 8) each frame has the same label
```

[55]:
```
torch.Size([9, 8])
```

[56]:
```
label
```

[56]:
```
tensor([[0., 0., 0., 0., 0., 1., 0., 0.],
        [0., 0., 0., 0., 0., 1., 0., 0.],
        [0., 0., 0., 0., 0., 1., 0., 0.],
        [0., 0., 0., 0., 0., 1., 0., 0.],
        [0., 0., 0., 0., 0., 1., 0., 0.],
        [0., 0., 0., 0., 0., 1., 0., 0.],
        [0., 0., 0., 0., 0., 1., 0., 0.],
        [0., 0., 0., 0., 0., 1., 0., 0.],
        [0., 0., 0., 0., 0., 1., 0., 0.]])
```

[57]:
```
clip.shape # (12, 9, C, H, W) 12 player , 9 frames, Channls , High, Width
```

[57]:
```
torch.Size([12, 9, 3, 224, 224])
```

[58]:
```
frame0 = clip[:, 0, : , : , :]

titles = [f"Player {i+1}" for i in range(12)]

plt.figure(figsize=(15, 5))
for i, player in enumerate(frame0):
    plt.subplot(2, 6, i + 1)
    plt.imshow(player.permute(1, 2, 0).numpy())
    plt.title(titles[i])
    plt.axis('off')

plt.tight_layout()
plt.show()
```

[59]:
```
frame1 = clip[:, 1, : , : , :]

plt.figure(figsize=(15, 5))
for i, player in enumerate(frame1):
    plt.subplot(2, 6, i + 1)
    plt.imshow(player.permute(1, 2, 0).numpy())
    plt.title(titles[i])
    plt.axis('off')

plt.tight_layout()
plt.show()
```

[60]:
```
frame2 = clip[:, 2, : , : , :]

plt.figure(figsize=(15, 5))
for i, player in enumerate(frame2):
    plt.subplot(2, 6, i + 1)
    plt.imshow(player.permute(1, 2, 0).numpy())
    plt.title(titles[i])
    plt.axis('off')

plt.tight_layout()
plt.show()
```

[61]:
```
frame3 = clip[:, 3, : , : , :]

plt.figure(figsize=(15, 5))
for i, player in enumerate(frame3):
    plt.subplot(2, 6, i + 1)
    plt.imshow(player.permute(1, 2, 0).numpy())
    plt.title(titles[i])
    plt.axis('off')

plt.tight_layout()
plt.show()
```

[62]:
```
frame4 = clip[:, 4, : , : , :]

plt.figure(figsize=(15, 5))
for i, player in enumerate(frame4):
    plt.subplot(2, 6, i + 1)
    plt.imshow(player.permute(1, 2, 0).numpy())
    plt.title(titles[i])
    plt.axis('off')
```

```
plt.tight_layout()
plt.show()
```

[63]:
```
frame5 = clip[:, 5, : , : , :]

plt.figure(figsize=(15, 5))
for i, player in enumerate(frame5):
    plt.subplot(2, 6, i + 1)
    plt.imshow(player.permute(1, 2, 0).numpy())
    plt.title(titles[i])
    plt.axis('off')

plt.tight_layout()
plt.show()
```

[64]:
```
frame6 = clip[:, 6, : , : , :]

plt.figure(figsize=(15, 5))
for i, player in enumerate(frame6):
```

```
    plt.subplot(2, 6, i + 1)
    plt.imshow(player.permute(1, 2, 0).numpy())
    plt.title(titles[i])
    plt.axis('off')

plt.tight_layout()
plt.show()
```

[65]:
```
frame7 = clip[:, 7, : , : , :]

plt.figure(figsize=(15, 5))
for i, player in enumerate(frame7):
    plt.subplot(2, 6, i + 1)
    plt.imshow(player.permute(1, 2, 0).numpy())
    plt.title(titles[i])
    plt.axis('off')

plt.tight_layout()
plt.show()
```

[66]:
```
frame8 = clip[:, 8, : , : , :]

plt.figure(figsize=(15, 5))
for i, player in enumerate(frame8):
    plt.subplot(2, 6, i + 1)
    plt.imshow(player.permute(1, 2, 0).numpy())
    plt.title(titles[i])
    plt.axis('off')

plt.tight_layout()
plt.show()
```

4.2 Can return a full clip with sorted player crop with its group label (all the same) ((12, 9, crop frame), tensor(9,8)) ***needed for B8*** **Note:** we sort player recored x-axis to separate each team.

[67]:
```
transforms = A.Compose([
    A.Resize(224, 224),
    ToTensorV2()
])

data_loader = Group_Activity_DataSet(videos_path, annot_path,␣
 ↪split=train_spilt, crops=True , seq=True, sort=True,␣
 ↪labels=group_activity_labels, transform=transforms)
```

[68]:
```
clip, label = data_loader[50]
clip.shape
```

[68]:
```
torch.Size([12, 9, 3, 224, 224])
```

[69]:
```
frist_frame = clip[:, 0, : , : , :] # take the frist frame

titles = [f"Player {i+1}" for i in range(12)]
```

```
plt.figure(figsize=(15, 5))

for i, player in enumerate(frist_frame[:6]): # take frist team
    plt.subplot(1, 6, i + 1)
    plt.imshow(player.permute(1, 2, 0).numpy())
    plt.title(titles[i])
    plt.axis('off')

plt.tight_layout()
plt.show()
```

[70]:
```
plt.figure(figsize=(15, 5))

for i, player in enumerate(frist_frame[6:]): # take second team
    plt.subplot(1, 6, i + 1)
    plt.imshow(player.permute(1, 2, 0).numpy())
    plt.title(titles[i + 6])
    plt.axis('off')

plt.tight_layout()
plt.show()
```

[71]:
```
clip, label = data_loader[600]

frist_frame = clip[:, 0, : , : , :] # take the frist frame

titles = [f"Player {i+1}" for i in range(12)]
```

```
plt.figure(figsize=(15, 5))

for i, player in enumerate(frist_frame[:6]): # take frist team
    plt.subplot(1, 6, i + 1)
    plt.imshow(player.permute(1, 2, 0).numpy())
    plt.title(titles[i])
    plt.axis('off')

plt.tight_layout()
plt.show()
```

[72]:
```
plt.figure(figsize=(15, 5))

for i, player in enumerate(frist_frame[6:]): # take second team
    plt.subplot(1, 6, i + 1)
    plt.imshow(player.permute(1, 2, 0).numpy())
    plt.title(titles[i + 6])
    plt.axis('off')

plt.tight_layout()
plt.show()
```