In [1]:
```
!git clone https://github.com/Sh-31/Group-Activity-Recognition.git
```

```
fatal: destination path 'Group-Activity-Recognition' already exists and is not an empty directory.
```

In [2]:
```
!pip install -U albumentations -q
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 80.1/80.1 kB 4.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 66.0/66.0 kB 2.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 274.9/274.9 kB 9.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 632.7/632.7 kB 20.6 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 303.5/303.5 kB 15.1 MB/s eta 0:00:00
```

In [3]:
```
### 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
#       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

#  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*.

# Hierarchical Group Activity Data Loader:
#   1. Can return crop of player in the same clip, each players label and group label of the clip ( (12, 9, crop frame), (12, 9, 9), (9,8) )

# 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= "/kaggle/working/Group-Activity-Recognition"
sys.path.append(os.path.abspath(PROJECT_ROOT))
from data_utils import Person_Activity_DataSet, Group_Activity_DataSet, Hierarchical_Group_Activity_DataSet
```

```
dataset_root = "/kaggle/input/group-activity-recognition-volleyball"
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)}

activities_labels = {"person": person_activity_labels, "group": group_activity_labels}

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]
```

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.*

In [4]:
```
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=trans
```

In [5]:
```
len(data_loader)
```

Out[5]: 231327

In [6]:
```
frame , label = data_loader[0]

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

Out[6]: torch.Size([9])

In [7]:
```
label
```

Out[7]: tensor([0., 0., 0., 0., 0., 0., 0., 0., 1.], dtype=torch.float64)

In [8]:
```
frame.shape # (C, H, W)
```

Out[8]: torch.Size([3, 224, 224])

In [9]:
```
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

In [10]:
```
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

In [11]:
```
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
```

In [12]:
```
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*.

In [13]:
```
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=transf
```

In [14]:
```
len(data_loader)
```

Out[14]: 2152

In [15]:
```
clip, label = data_loader[100]

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

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

In [16]:
```
label[0, 0, :]
```

Out[16]: tensor([0., 0., 0., 0., 0., 0., 0., 0., 1.], dtype=torch.float64)

In [17]:
```
clip.shape #  (12 player, 9 frame, C, H, W)
```

Out[17]: torch.Size([12, 9, 3, 224, 224])

In [18]:
```
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

In [19]:
```
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
```

In [20]:
```
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
```

In [21]:
```
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

In [22]:
```
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

In [23]:
```
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

In [24]:
```
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

In [25]:
```
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

In [26]:
```
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

Test group activity data loader

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

In [27]:
```
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, tra
```

In [28]:
```
len(data_loader)
```

Out[28]: 19368

In [29]:
```
frame , label = data_loader[0]
```

In [30]:
```
print(label.shape) # (,8)
label
```

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

Out[30]: tensor([0., 0., 0., 0., 0., 1., 0., 0.])

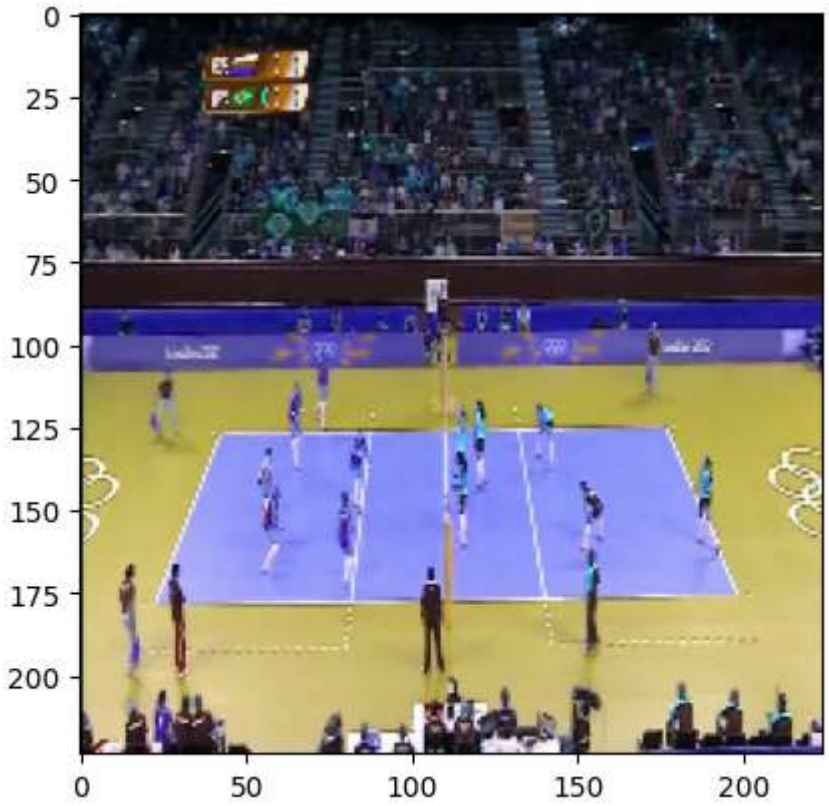
In [31]:
```
frame.shape # (C, H , W)
```

Out[31]: torch.Size([3, 224, 224])

In [32]:
```
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()
```

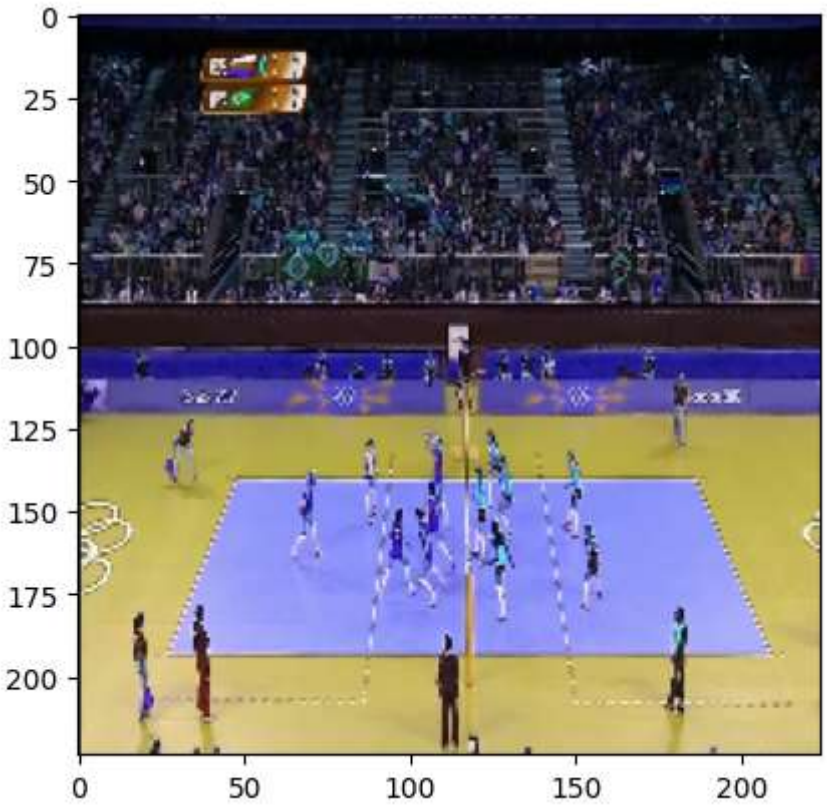
```
l-pass
```



In [33]:
```
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.*

In [34]:
```
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, trans
```

In [35]:
```
len(data_loader) # the differents between case 1 and 2 the input consist of 12 bbox
```

Out[35]: 19368

In [36]:
```
frame_crops, label = data_loader[152]
```

In [37]:
```
print(label.shape) # (,8)
label
```

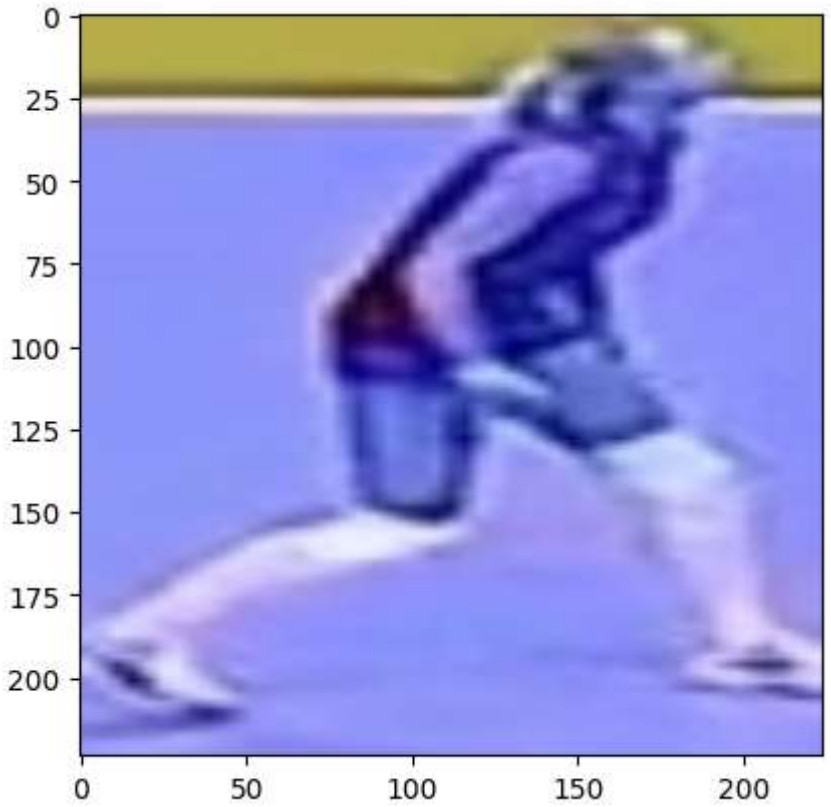
```
torch.Size([8])
```

Out[37]:
```
tensor([0., 0., 0., 0., 0., 1., 0., 0.])
```

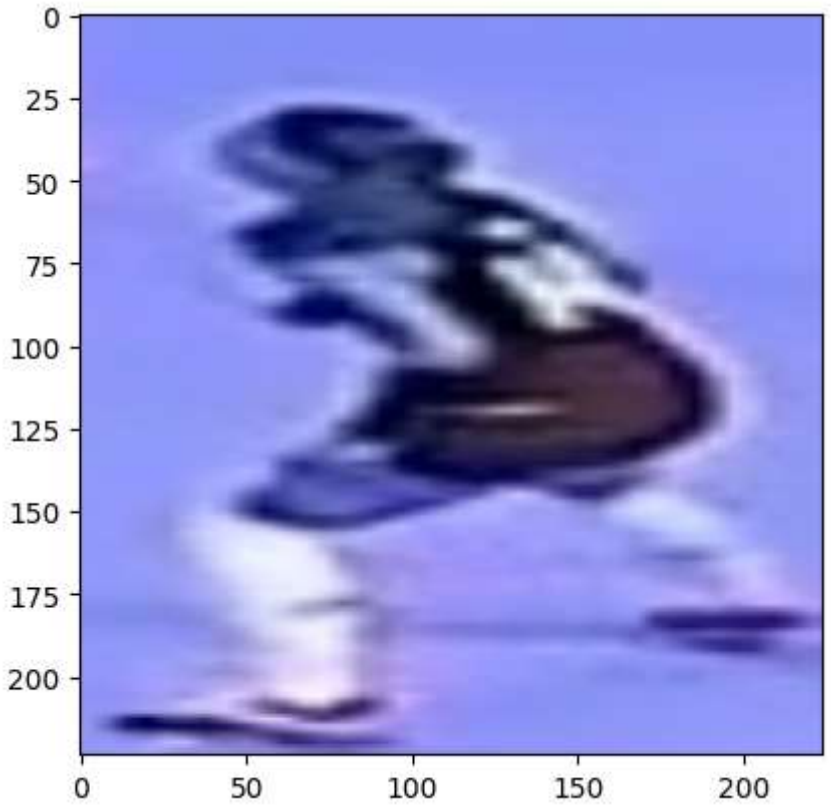
In [38]:
```
frame_crops.shape # (12, C, H, W) ---> 12 bbox of the frame
```

Out[38]:
```
torch.Size([12, 3, 224, 224])
```

In [39]:
```
plt.imshow(frame_crops[0].permute(1,2,0))  # first bbox of the fraem
plt.show()
```



In [40]:
```
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*.

In [41]:
```
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, tran
```

In [42]:
```
len(data_loader)
```

Out[42]: 2152

In [43]:
```
clip , label = data_loader[100]
clip.shape # (9 frames, C, H, W)
```

Out[43]: torch.Size([9, 3, 224, 224])

In [44]:
```
group_activity_labels
```

Out[44]:
```
{'r_set': 0,
 'r_spike': 1,
 'r-pass': 2,
 'r_winpoint': 3,
 'l_winpoint': 4,
 'l-pass': 5,
 'l-spike': 6,
 'l_set': 7}
```

In [45]:
```
plt.figure(figsize=(2, 2)) # First frame of the clip
plt.imshow(clip[0].permute(1, 2, 0).numpy())
plt.axis('off')
plt.show()
```

In [46]:
```
plt.figure(figsize=(2, 2)) # second frame of the clip
plt.imshow(clip[1].permute(1, 2, 0).numpy())
plt.axis('off')
plt.show()
```

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

In [48]:
```
plt.figure(figsize=(2, 2))
plt.imshow(clip[3].permute(1, 2, 0).numpy())
plt.axis('off')
plt.show()
```

In [49]:
```
plt.figure(figsize=(2, 2))
plt.imshow(clip[4].permute(1, 2, 0).numpy())
plt.axis('off')
plt.show()
```

In [50]:
```
plt.figure(figsize=(2, 2))
plt.imshow(clip[5].permute(1, 2, 0).numpy())
```

```
plt.axis('off')
plt.show()
```

In [51]:
```
plt.figure(figsize=(2, 2))
plt.imshow(clip[6].permute(1, 2, 0).numpy())
plt.axis('off')
plt.show()
```

In [52]:
```
plt.figure(figsize=(2, 2))
plt.imshow(clip[7].permute(1, 2, 0).numpy())
plt.axis('off')
plt.show()
```

In [53]:
```
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*.

In [54]:
```
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, trans
```

In [55]:
```
len(data_loader)
```

Out[55]: 2152

In [56]:
```
clip, label = data_loader[50]
```

In [57]:
```
label.shape # (9, 8) each frame has the same label
```

Out[57]: torch.Size([9, 8])

In [58]:
```
label
```

Out[58]:
```
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.]])
```

In [59]:
```
clip.shape # (12, 9, C, H, W) 12 player , 9 frames, Channls , High, Width
```

Out[59]:
```
torch.Size([12, 9, 3, 224, 224])
```

In [60]:
```
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()
```

In [61]:

```
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()
```

In [62]:

```
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()
```

In [63]:

```
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()
```

In [64]:

```
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()
```

In [65]:
```
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()
```

In [66]:

```
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()
```

In [67]:
```
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()
```

In [68]:

```
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.

In [69]:
```
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_lal
```

In [70]:
```
clip, label = data_loader[50]
clip.shape
```

Out[70]:
```
torch.Size([12, 9, 3, 224, 224])
```

In [71]:
```
label_idex = label[0].argmax().item() # group label at frist frame
group_label = group_activity_clases[label_idex]
print(f"Group activity label: {group_label}")
```

```
Group activity label: l-pass
```

In [72]:
```
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()
```

In [73]:
```
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()
```

In [74]:
```
clip, label = data_loader[600]
label_idex = label[0].argmax().item() # group label at frist frame
group_label = group_activity_clases[label_idex]

print(f"Group activity label: {group_label}")
```

```
Group activity label: r_spike
```

In [75]:
```
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()
```

In [76]:
```
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()
```

In [77]:
```
del data_loader
```

Test hierarchical group activity dataSet

1. Can return crop of player in the same clip, each players label and group label of the clip ((12, 9, crop frame), (12, 9, 9), (9,8)) *needed for B9*.

In [78]:
```
transforms = A.Compose([
    A.Resize(224, 224),
    ToTensorV2()
])

data_loader = Hierarchical_Group_Activity_DataSet(
    videos_path=videos_path,
    annot_path=annot_path,
    split=train_spilt,
    labels=activities_labels,
    transform=transforms
)
```

In [79]:
```
len(data_loader)
```

Out[79]: 2152

In [80]:
```
clip, person_labels, group_labels = data_loader[600]
clip.shape # (12, 9, 3, 224, 224) => 12 player, 9 frame, c, h, w
```

Out[80]: torch.Size([12, 9, 3, 224, 224])

In [81]:
```
person_labels.shape # (12, 9, 9) => each player at each frame label
```

Out[81]: torch.Size([12, 9, 9])

In [82]:
```
group_labels.shape # (9, 8) => group label at each frame
```

Out[82]: torch.Size([9, 8])

In [83]:
```
label_idex = group_labels[0].argmax().item() # group label at frist frame
group_label = group_activity_clases[label_idex]
print(f"Group activity label: {group_label}")
```

```
Group activity label: r_spike
```

In [84]:
```
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

    label_idex = person_labels[i, 0].argmax().item() # take label of the player at frist frame
    player_label = people_activity_clases[label_idex]

    plt.subplot(1, 6, i + 1)
    plt.imshow(player.permute(1, 2, 0).numpy())
    plt.title(f"{titles[i]} - label: {player_label}")
    plt.axis('off')

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

In [85]:
```
plt.figure(figsize=(15, 5))

for i, player in enumerate(frist_frame[6:]): # take second team
    label_idex = person_labels[i, 0].argmax().item() # take label of the player at frist frame
    player_label = people_activity_clases[label_idex]

    plt.subplot(1, 6, i + 1)
    plt.imshow(player.permute(1, 2, 0).numpy())
    plt.title(f"{titles[i]} - label: {player_label}")
    plt.axis('off')

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