



# Proposal for Supporting Optional Overlays with MV-HEVC

**Niko Stefanoski, Oliver Wang,  
Aljosa Smolic, Ted Szypulski**

Disney Research Zurich  
ESPN



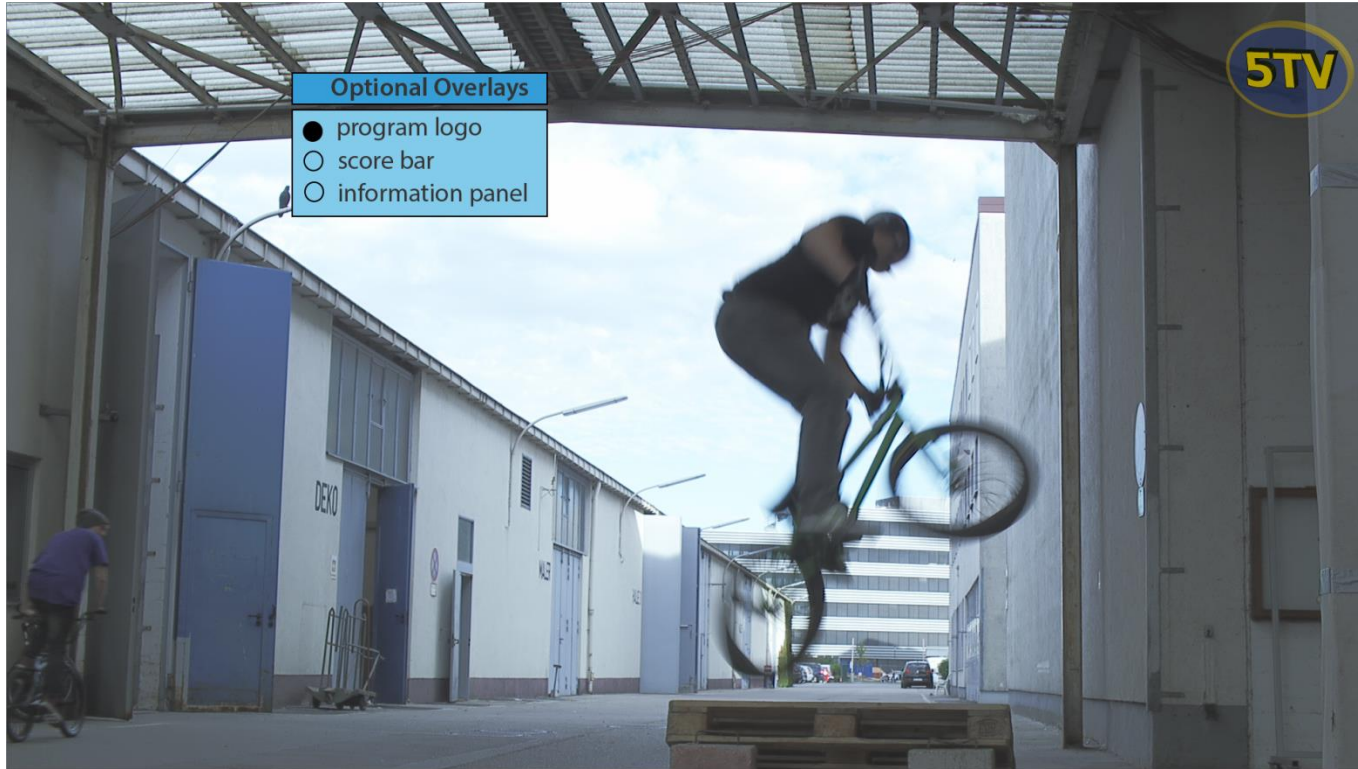


# What are Optional Overlays?





# What are Optional Overlays?





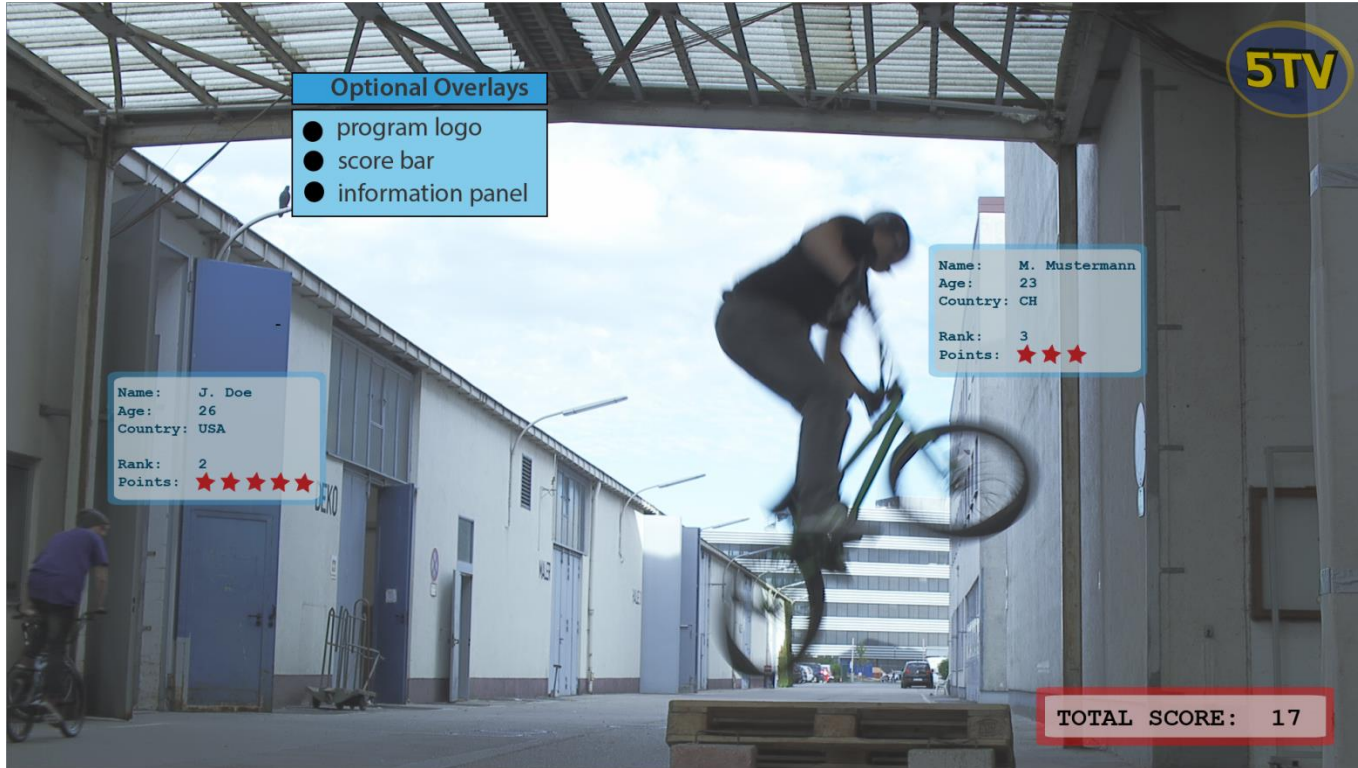
# What are Optional Overlays?







# What are Optional Overlays?



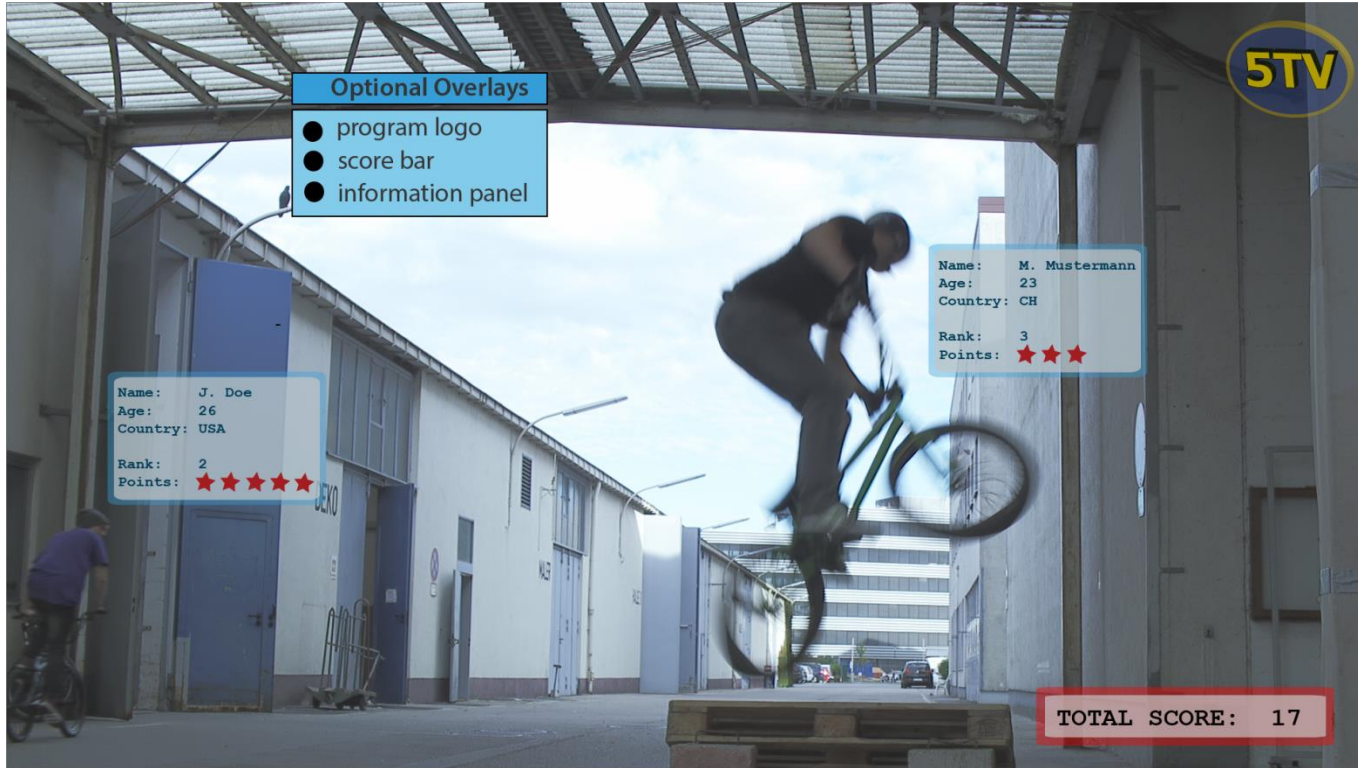


# What are Optional Overlays?





# What are Optional Overlays?







# What are Optional Overlays?







# What are Optional Overlays?





# Previous Solutions

- Different distribution channels for video and overlay data
  - e.g. video through cable network
  - e.g. overlays through Internet
  - temporal synchronization problems
- Overlays represented by meta data not pixel data
  - rendering of overlays at the receiver required
  - no control over visual appearance
- Proprietary solutions





# Proposal

- Realize Optional Overlays functionality with help of MV-HEVC
- Represent 2D video and overlay data with MV-HEVC
  - 2D Video Data represented in base view
  - Overlay Data represented supplemental views
    - overlay data can be temporally changing
- Define SEI message to
  - signal availability of Optional Overlays functionality
  - provide identifiers for overlay elements





# Video and Overlay Data

## 2D Video Data

video pictures, YUV

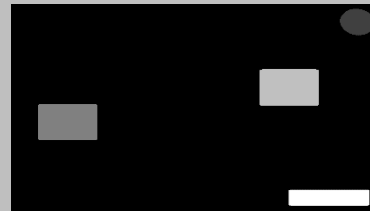


## Overlay Data

overlay pictures, YUV



label pictures, Y



alpha matte pictures, Y







# SEI Message Syntax and Semantics

	Descriptor
optional_overlays_info ( payloadSize ) {	
<b>oov_info_cancel_flag</b>	u(1)
if( oov_info_cancel_flag == 0 ) {	
<b>oov_setup_id</b>	u(3)
<b>oov_ol_views_cnt_minus1</b>	ue(v)
<b>oov_ol_elem_cnt_minus1</b>	ue(v)
for ( i=0; i <= oov_ol_elem_cnt_minus1; i++ ) {	
for ( j=0; j <= oov_ol_views_cnt_minus1; j++ )	
<b>oov_ol_elem_name[i][j]</b>	f(512)
<b>oov_label_id[i]</b>	u(v)
}	
<b>oov_label_offset</b>	u(v)
if ( oov_setup_id == 0    oov_setup_id == 1 )	
<b>oov_alpha_present_flag</b>	u(1)
}	
}	

Specifies  
persistence of SEI  
message





# SEI Message Syntax and Semantics

optional_overlays_info ( payloadSize ) {	<b>Descriptor</b>
<b>oov_info_cancel_flag</b>	u(1)
if( oov_info_cancel_flag == 0 ) {	
<b>oov_setup_id</b>	u(3)
<b>oov_ol_views_cnt_minus1</b>	ue(v)
<b>oov_ol_elem_cnt_minus1</b>	ue(v)
for ( i=0; i <= oov_ol_elem_cnt_minus1; i++ ) {	
for ( j=0; j <= oov_ol_views_cnt_minus1; j++ )	
<b>oov_ol_elem_name[i][j]</b>	f(512)
<b>oov_label_id[i]</b>	u(v)
}	
<b>oov_label_offset</b>	u(v)
if ( oov_setup_id == 0    oov_setup_id == 1 )	
<b>oov_alpha_present_flag</b>	u(1)
}	
}	

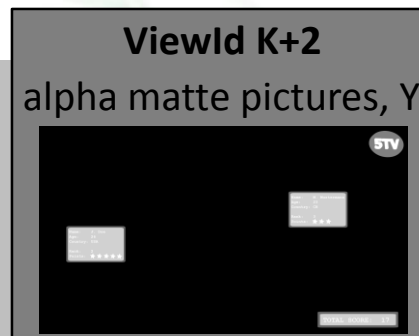
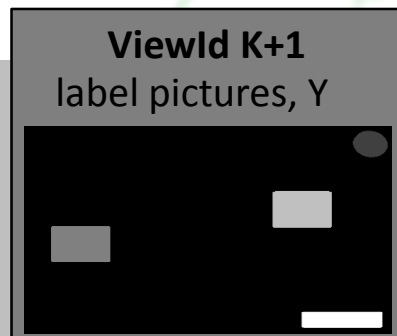
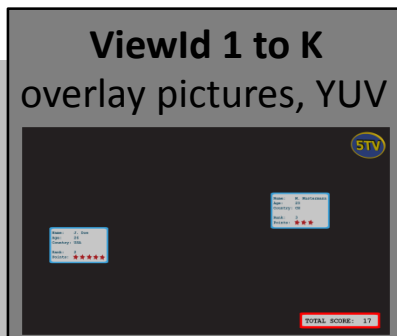
Value	Description
0	Setup A
1	Setup B
2	Setup C
3-15	reserved





# Video and Overlay Data

## Setup A



Number of views:  $\geq 4$





# Video and Overlay Data

## Setup B

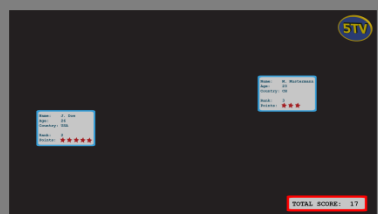
### ViewId 0

video pictures, YUV



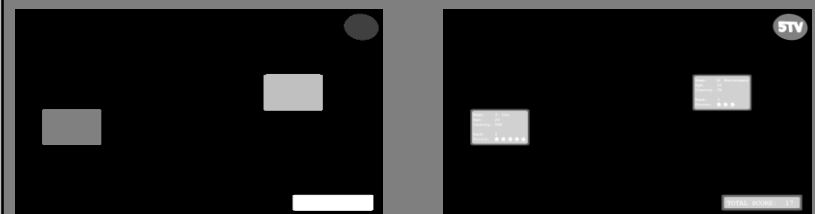
### ViewId 1 to K

overlay pictures, YUV



### ViewId K+1

label + alpha matte pictures, YUV



Number of views:  $\geq 3$







# Video and Overlay Data

## Setup C

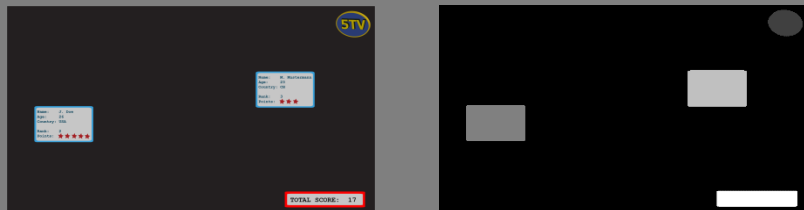
### ViewId 0

video pictures, YUV



### ViewId 1

overlay + label pictures, YUVA



Number of views: 2

Auxiliary pictures required!



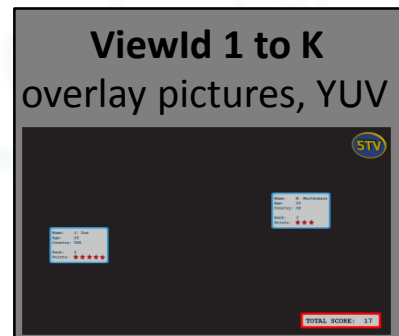


# SEI Message Syntax and Semantics

optional_overlays_info ( payloadSize ) {	<b>Descriptor</b>
oov_info_cancel_flag	u(1)
if( oov_info_cancel_flag == 0 ) {	
oov_setup_id	u(3)
oov_ol_views_cnt_minus1	ue(v)
oov_ol_elem_cnt_minus1	ue(v)
for ( i=0; i <= oov_ol_elem_cnt_minus1; i++ ) {	
for ( j=0; j <= oov_ol_views_cnt_minus1; j++ )	
oov_ol_elem_name[i][j]	f(512)
oov_label_id[i]	u(v)
}	
oov_label_offset	u(v)
if ( oov_setup_id == 0    oov_setup_id == 1 )	
oov_alpha_present_flag	u(1)
}	
}	

Number of views K  
for overlay pictures  
minus 1

Example:



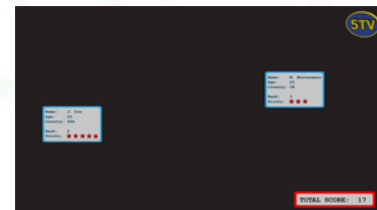


# SEI Message Syntax and Semantics

optional_overlays_info ( payloadSize ) {	<b>Descriptor</b>
<b>oov_info_cancel_flag</b>	u(1)
if( oov_info_cancel_flag == 0 ) {	
<b>oov_setup_id</b>	u(3)
<b>oov_ol_views_cnt_minus1</b>	ue(v)
<b>oov_ol_elem_cnt_minus1</b>	ue(v)
for ( i=0; i <= oov_ol_elem_cnt_minus1; i++ ) {	
for ( j=0; j <= oov_ol_views_cnt_minus1; j++ )	
<b>oov_ol_elem_name[i][j]</b>	f(512)
<b>oov_label_id[i]</b>	u(v)
}	
<b>oov_label_offset</b>	u(v)
if ( oov_setup_id == 0    oov_setup_id == 1 )	
<b>oov_alpha_present_flag</b>	u(1)
}	
}	

Number of overlay elements within a overlay picture minus 1

Example:





# SEI Message Syntax and Semantics

	Descriptor
optional_overlays_info ( payloadSize ) {	
<b>oov_info_cancel_flag</b>	u(1)
if( oov_info_cancel_flag == 0 ) {	
<b>oov_setup_id</b>	u(3)
<b>oov_ol_views_cnt_minus1</b>	ue(v)
<b>oov_ol_elem_cnt_minus1</b>	ue(v)
for ( i=0; i <= oov_ol_elem_cnt_minus1; i++ ) {	
for ( j=0; j <= oov_ol_views_cnt_minus1; j++ )	
<b>oov_ol_elem_name[i][j]</b>	f(512)
<b>oov_label_id[i]</b>	u(v)
}	
<b>oov_label_offset</b>	u(v)
if ( oov_setup_id == 0    oov_setup_id == 1 )	
<b>oov_alpha_present_flag</b>	u(1)
}	
}	

Names for all  
overlay elements in  
all overlay pictures

Example

## Optional Overlays

- ☐ program logo
- ☐ score bar
- ☐ information panel







# SEI Message Syntax and Semantics

optional_overlays_info ( payloadSize ) {	<b>Descriptor</b>
oov_info_cancel_flag	u(1)
if( oov_info_cancel_flag == 0 ) {	
oov_setup_id	u(3)
oov_ol_views_cnt_minus1	ue(v)
oov_ol_elem_cnt_minus1	ue(v)
for ( i=0; i <= oov_ol_elem_cnt_minus1; i++ ) {	
for ( j=0; j <= oov_ol_views_cnt_minus1; j++ )	
oov_ol_elem_name[i][j]	f(512)
oov_label_id[i]	u(v)
}	
oov_label_offset	u(v)
if ( oov_setup_id == 0    oov_setup_id == 1 )	
oov_alpha_present_flag	u(1)
}	
}	

Numerical identifiers for overlay elements within a label picture

Example:





# SEI Message Syntax and Semantics

optional_overlays_info ( payloadSize ) {	<b>Descriptor</b>
<b>oov_info_cancel_flag</b>	u(1)
if( oov_info_cancel_flag == 0 ) {	
<b>oov_setup_id</b>	u(3)
<b>oov_ol_views_cnt_minus1</b>	ue(v)
<b>oov_ol_elem_cnt_minus1</b>	ue(v)
for ( i=0; i <= oov_ol_elem_cnt_minus1; i++ ) {	
for ( j=0; j <= oov_ol_views_cnt_minus1; j++ )	
<b>oov_ol_elem_name[i][j]</b>	f(512)
<b>oov_label_id[i]</b>	u(v)
}	
<b>oov_label_offset</b>	u(v)
if ( oov_setup_id == 0    oov_setup_id == 1 )	
<b>oov_alpha_present_flag</b>	u(1)
}	
}	

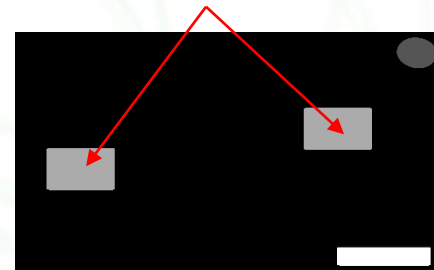
Used to specify a range of label values which identify an overlay element

Example:

oov\_label\_offset = 32

oov\_label\_id = 170

$$170-32 \leq x \leq 170+32$$





# SEI Message Syntax and Semantics

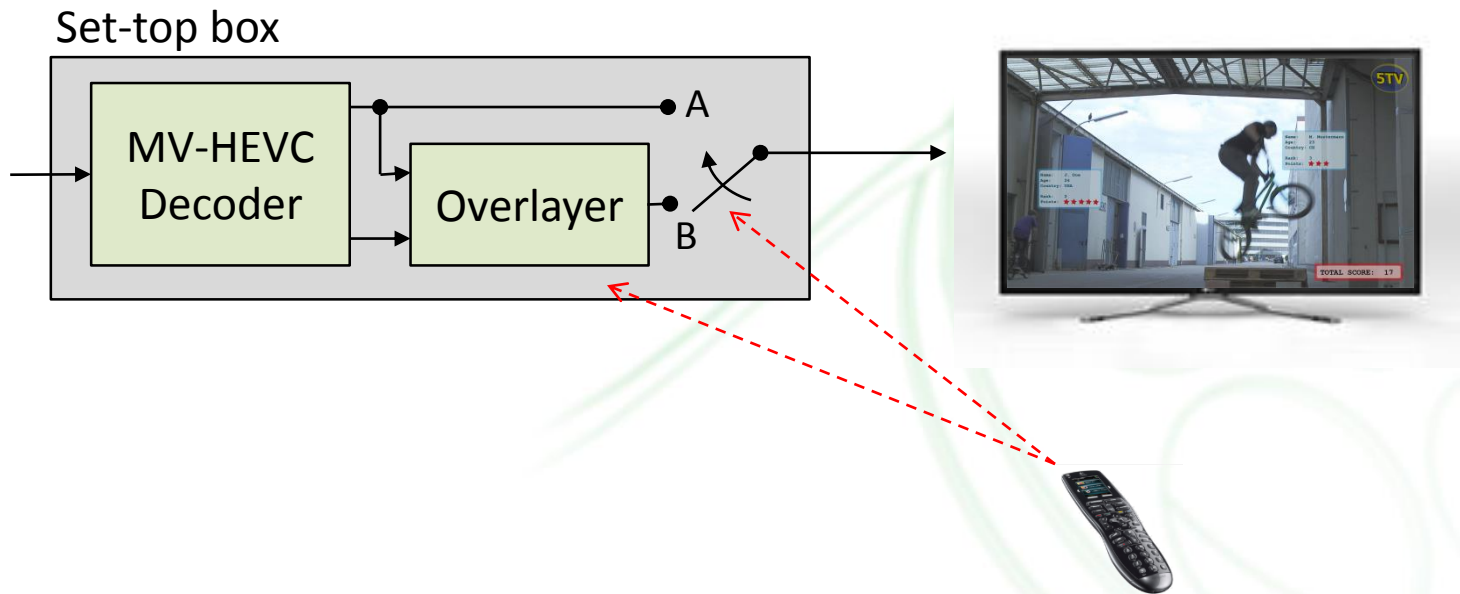
<b>optional_overlays_info ( payloadSize ) {</b>	<b>Descriptor</b>
<b>oov_info_cancel_flag</b>	u(1)
if( oov_info_cancel_flag == 0 ) {	
<b>oov_setup_id</b>	u(3)
<b>oov_ol_views_cnt_minus1</b>	ue(v)
<b>oov_ol_elem_cnt_minus1</b>	ue(v)
for ( i=0; i <= oov_ol_elem_cnt_minus1; i++ ) {	
for ( j=0; j <= oov_ol_views_cnt_minus1; j++ )	
<b>oov_ol_elem_name[i][j]</b>	f(512)
<b>oov_label_id[i]</b>	u(v)
}	
<b>oov_label_offset</b>	u(v)
if ( oov_setup_id == 0    oov_setup_id == 1 )	
<b>oov_alpha_present_flag</b>	u(1)
}	
}	

Specifies if alpha  
matte pictures are  
present





# Optional Overlays System



A – video without overlay  
B – video with overlay





# User Selection

## Data decoded by MV-HEVC:

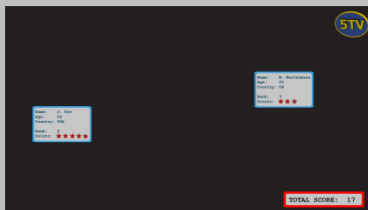
2D Video Data

resSamples<sub>Vid</sub>

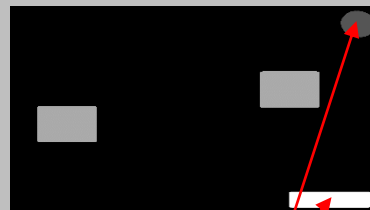


Overlay Data

resSamples<sub>OL</sub>



resSamples<sub>Label</sub>



resSamples<sub>Alpha</sub>



## Data specified by user selection:

### Optional Overlays

- program logo
- score bar
- information panel



user\_oov\_label\_cnt = 2  
user\_oov\_label\_id[0] = 85  
user\_oov\_label\_id[1] = 255  
user\_oov\_ov\_view\_id[0] = 0





# Derivation Process

**Input:** decoded pictures, SEI message syntax elements, data from user selection

**Output:** picture with overlays selected by the user

```
for( y=0; y <= pic_height_in_luma_samples; y++ ) {  
    for( x=0; x <= pic_width_in_luma_samples; x++ ) {  
        alpha=0  
        ov_view_id_cur = 0  
        for( i=0; i <= user_oov_label_cnt; i++ ) {  
            if ( ( user_oov_label_id[i] - oov_label_offset) <= resSamples_Label[x][y] &&  
                (user_oov_label_id[i] + oov_label_offset) >= resSamples_Label[x][y] ) {  
                alpha = 1  
                ov_view_id_cur = user_oov_ov_view_id[i]  
            }  
        }  
        if ( oov_alpha_present_flag == 1 && alpha == 1 )  
            alpha = resSamples_Alpha[x][y] ÷ ( (1 << BitDepthy) - 1 )  
        resSamples_VidOut[x][y] = ( 1 - alpha ) * resSamples_Vid[x][y] + alpha * resSamples_Ol[ov_view_id_cur][x][y]  
    }  
}
```





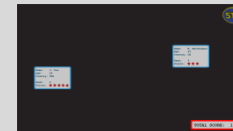


# Derivation Process

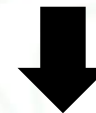
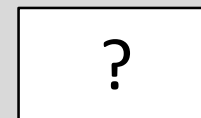
**Input:** decoded pictures, SEI message syntax elements, data from user selection

**Output:** picture with overlays selected by the user

```
for( y=0; y <= pic_height_in_luma_samples; y++ ) {  
    for( x=0; x <= pic_width_in_luma_samples; x++ ) {  
        alpha=0  
        ov_view_id_cur = 0  
        for( i=0; i <= user_oov_label_cnt; i++ ) {  
            if ( ( user_oov_label_id[i] - oov_label_offset) <= resSamples_Label[x][y] &&  
                (user_oov_label_id[i] + oov_label_offset) >= resSamples_Label[x][y] ) {  
                alpha = 1  
                ov_view_id_cur = user_oov_ov_view_id[i]  
            }  
        }  
        if ( oov_alpha_present_flag == 1 && alpha == 1 )  
            alpha = resSamples_Alpha[x][y] ÷ ( (1 << BitDepthy) - 1 )  
        resSamples_VidOut[x][y] = ( 1 - alpha ) * resSamples_Vid[x][y] + alpha * resSamples_OL[ov_view_id_cur][x][y]  
    }  
}
```



alpha matte



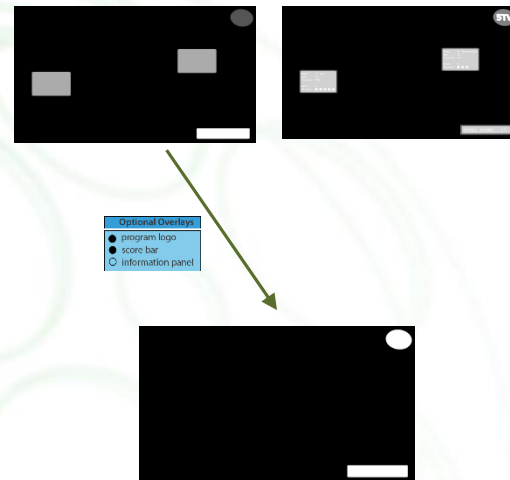


# Derivation Process

**Input:** decoded pictures, SEI message syntax elements, data from user selection

**Output:** picture with overlays selected by the user

```
for( y=0; y <= pic_height_in_luma_samples; y++ ) {  
    for( x=0; x <= pic_width_in_luma_samples; x++ ) {  
  
        alpha=0  
        ov_view_id_cur = 0  
        for( i=0; i <= user_oov_label_cnt; i++ ) {  
            if ( ( user_oov_label_id[i] - oov_label_offset) <= resSamples_Label[x][y] &&  
                (user_oov_label_id[i] + oov_label_offset) >= resSamples_Label[x][y] ) {  
                alpha = 1  
                ov_view_id_cur = user_oov_ov_view_id[i]  
            }  
        }  
  
        if ( oov_alpha_present_flag == 1 && alpha == 1 )  
            alpha = resSamples_Alpha[x][y] ÷ ( (1 << BitDepthy) - 1 )  
        resSamples_VidOut[x][y] = ( 1 - alpha ) * resSamples_Vid[x][y] + alpha * resSamples_Ol[ov_view_id_cur][x][y]  
    }  
}
```



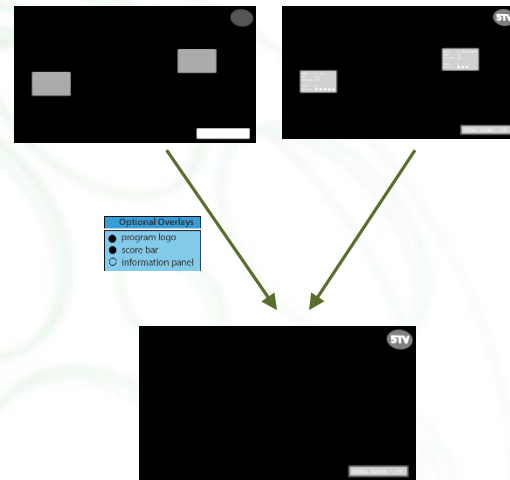


# Derivation Process

**Input:** decoded pictures, SEI message syntax elements, data from user selection

**Output:** picture with overlays selected by the user

```
for( y=0; y <= pic_height_in_luma_samples; y++ ) {  
    for( x=0; x <= pic_width_in_luma_samples; x++ ) {  
        alpha=0  
        ov_view_id_cur = 0  
        for( i=0; i <= user_oov_label_cnt; i++ ) {  
            if ( ( user_oov_label_id[i] - oov_label_offset) <= resSamples_Label[x][y] &&  
                (user_oov_label_id[i] + oov_label_offset) >= resSamples_Label[x][y] ) {  
                alpha = 1  
                ov_view_id_cur = user_oov_ov_view_id[i]  
            }  
        }  
        if ( oov_alpha_present_flag == 1 && alpha == 1 )  
            alpha = resSamples_Alpha[x][y] ÷ ( (1 << BitDepthy) - 1 )  
        resSamples_VidOut[x][y] = ( 1 - alpha ) * resSamples_Vid[x][y] + alpha * resSamples_Ol[ov_view_id_cur][x][y]  
    }  
}
```



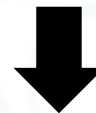
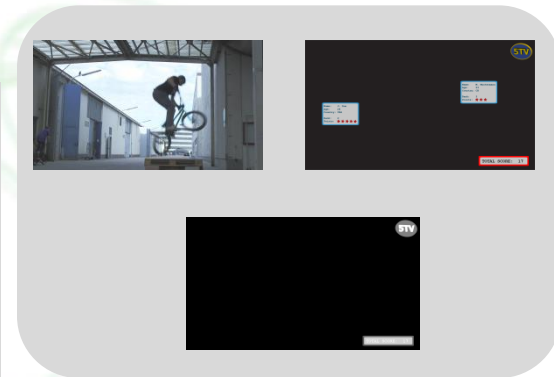


# Derivation Process

**Input:** decoded pictures, SEI message syntax elements, data from user selection

**Output:** picture with overlays selected by the user

```
for( y=0; y <= pic_height_in_luma_samples; y++ ) {
    for( x=0; x <= pic_width_in_luma_samples; x++ ) {
        alpha=0
        ov_view_id_cur = 0
        for( i=0; i <= user_oov_label_cnt; i++ ) {
            if ( ( user_oov_label_id[i] - oov_label_offset) <= resSamples_Label[x][y] &&
                (user_oov_label_id[i] + oov_label_offset) >= resSamples_Label[x][y] ) {
                alpha = 1
                ov_view_id_cur = user_oov_ov_view_id[i]
            }
        }
        if ( oov_alpha_present_flag == 1 && alpha == 1 )
            alpha = resSamples_Alpha[x][y] ÷ ( (1 << BitDepthy) - 1 )
        resSamples_VidOut[x][y] = ( 1 - alpha ) * resSamples_Vid[x][y] + alpha * resSamples_OL[ov_view_id_cur][x][y]
    }
}
```





# Summary and Conclusions

- Optional Overlays realized with MV-HEVC
  - International standard vs proprietary solutions
  - Overlays and video content are temporally synchronized (live distribution, storage)
  - Content producers/distributers can control visual appearance of overlays
- Wider exploitation of MV-HEVC technology
- Applications in sports broadcast, news and entertainment programs, advertisements, etc.





# Recommendation

Adopt the Optional Overlays SEI message in the MV-HEVC specification

